LA BIBLE DU GRAPHISME

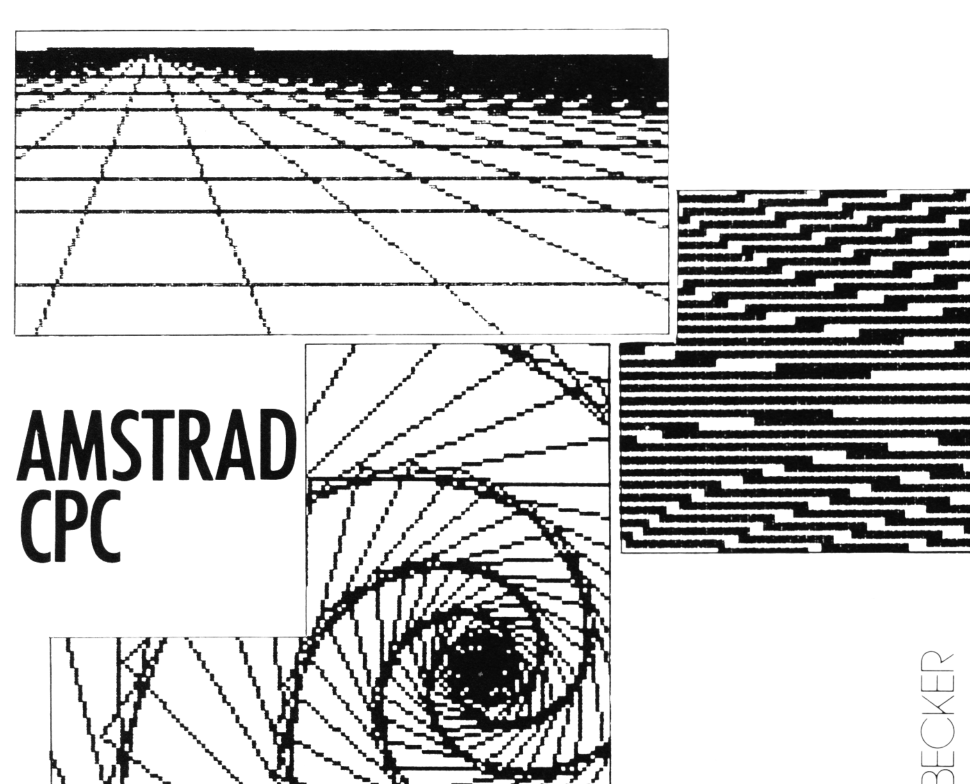
AMSTRAD CPC



EDITIONS MICRO APPLICATION

UN LIVRE DATA BECKER

LA BIBLE DU GRAPHISME



AMSTRAD CPC

EDITIONS MICRO APPLICATION

UN LIVRE DATA BECKER

Distribué par : MICRO APPLICATION
13, Rue Sainte Cécile
75009 PARIS

et

EDITIONS RADIO
189, Rue Saint Jacques
75005 PARIS

ISBN : 2-86899-090-8

Traduction Française assurée par Pascal HAUSMANN

Collection dirigée par Mr Philippe OLIVIER
Edition réalisée par Frédérique BEAUDONNET

Sommaire

1. **Introduction** 1

2. **Point, virgule, trait...** 5

2.1 Le premier point apparaît 6
2.2 Du point à la ligne 11
2.3 Formes géométriques simples 13
2.4 La couleur entre en jeu 18
2.5 Un petit programme de dessin 19

3. **Le jeu de caractères - une source d'éléments graphiques** 33

3.1 Le jeu de caractères du CPC 36
3.2 Caractères de commande BASIC 52
3.3 Plusieurs caractères forment une image 63
3.4 L'éditeur de jeu de caractères 69
3.5 Les accents 85
3.6 Un jeu de caractères alternatif 89
3.7 DESTROYED - Un jeu d'arcade 115
3.7.1 L'idée de jeu 116
3.7.2 Réalisation des scénarios 118
3.7.3 Animation 120

4. **Art abstrait en BASIC** 131

4.1 "Touche pas à mes cercles!" 133
4.2 Corne 136
4.3 Vague 138
4.3 Profondeur 140
4.4 Modèle d'interférence 142
4.5 Gridrunner 144

5. **Un bon croquis vaut mieux qu'un long discours - Les graphiques d'entreprise** 147

5.1 Le graphique de points 149
5.2 CPC-Chart 160
5.3 Graphique camembert 176

6. Représentation graphique de fonctions mathématiques 187

6.1 Le système de coordonnées 190
6.2 Fonctions linéaires 196
6.3 Fonctions quadratiques 200
6.4 Le plotter de fonction 203

7. Le graphisme en trois dimensions 215

7.1 Le système de coordonnées tridimensionnel 216
7.2 La projection parallèle 220
7.2.1 Rotation dans l'espace 225
7.3 Le projection centrale 226
7.4 Astuces pour une animation d'objet plus rapide 228
7.5 CPCs-World 231
7.5.1 Améliorations du programme 255

8. Allons plus loin - périphérique 259

8.1 Joystick 259
8.2 Le crayon optique 262
8.3 Copie d'écran graphique 267
8.4 Bande-texte 273

9. La programmation en langage machine sur le CPC 279

9.1 Le coeur du CPC - l'unité centrale Z80 279
9.2 Le contrôleur vidéo 285
9.3 Le Gate Array 289
9.4 Programmation graphique du Z80 292
9.5 Les principales routines de la ROM 305
9.6 Le listing de la ROM 333
9.6.1 Désassembleur Z80 pour le CPC 334
9.6.2 SCREEN PACK (SCR) 340
9.6.3 Text Screen (TXT) 373
9.6.4 Graphics Screen (GRA) 409
9.7 La puissance du langage machine 424
9.7.1 Les Sprites sur le CPC 424
9.7.2 Organisation de plusieurs pages écran 445
9.7.3 Scrolling 449

10. GSX - L'extension système graphique **455**

10.1 Qu'est-ce que GSX? 455
10.2 Implantation de GSX 458
10.3 Transmission de paramètres 463
10.4 Le jeu d'instructions GSX 469
10.5 GDOS de l'intérieur 491
10.6 Messages d'erreur de GSX 506
10.7 Spécifications de processeur de périphérique 507

Annexe **513**

Instructions BASIC ayant trait au graphisme 513

Lexique **527**

Index **535**

Avertissement : Pour des raisons techniques, le symbole flèche vers le haut sera remplacé dans cet ouvrage par un accent circonflexe (^).

1. Introduction

L'évolution du graphisme sur ordinateur

Le graphisme informatique à base de grilles est devenu aujourd'hui quelque chose de parfaitement naturel. Il est utilisé par exemple de plus en plus souvent dans des films, dans des spots publicitaires, par toutes sortes de sociétés ou même par des amateurs d'informatique. C'est pourquoi nombreux sont ceux, parmi ceux qui achètent de nos jours un ordinateur graphique, qui ne peuvent imaginer qu'il y a quelques années encore le graphisme sur ordinateur relevait encore du domaine des gros ordinateurs. Pour vous donner une idée du progrès vraiment exponentiel qui caractérise ce domaine, voici une description dans ses grandes lignes de l'évolution du graphisme sur ordinateur.

En 1966, les meilleurs ordinateurs disponibles disposaient d'une capacité mémoire d'environ 16 K octets. Si vous comparez à un CPC 464, vous voyez tout de suite que ce dernier dispose aujourd'hui d'une mémoire de travail quadruple, c'est-à-dire de 64 K octets. 16 K entiers sont d'ailleurs entièrement consacrés à la mémoire écran responsable du graphisme. On ne pouvait autrefois réserver comme mémoire écran que de très petites zones de la mémoire de travail, pour ne pas épuiser totalement la place mémoire disponible. Mais comme c'est justement la taille de la mémoire écran qui détermine la finesse des dessins sur ordinateur, on ne pouvait de ce fait dessiner que des images très grossières. On finit donc par décider de renoncer tout à fait à la sortie sur écran et de n'utiliser que l'imprimante comme machine de sortie des données.

On a développé pour les applications graphiques ce qu'on appelle le système de graphisme vectoriel. Cela consistait à dessiner directement des lignes sur un tube d'écran recouvert d'une épaisse couche de phosphore au moyen d'un rayon cathodique. L'ordinateur lui-même n'avait à produire qu'un nombre très réduit d'informations qui se limitaient simplement à la commande du rayon d'électrons. Comme l'écran continuait de briller pendant une heure du fait de l'épaisse couche de phosphore, l'ordinateur pouvait être utilisé à d'autres tâches pendant ce temps.

Cette méthode se révéla cependant vite très peu pratique car tout effacement prématuré de l'image provoquait un éclair désagréable qui empêchait de réaliser des dessins animés de mouvements continus. Ce problème tracassait beaucoup les experts en graphisme sur ordinateur de l'époque et c'est ainsi que furent bientôt utilisés des tubes d'image avec régénération. Ces tubes renouvelaient l'image 50 fois par seconde, permettant ainsi d'obtenir une image qui ne tremblote pas. L'image ne garde toutefois son apparence stable que si les vecteurs (les lignes) ne dépassent pas une longueur déterminée car il devient sinon impossible d'éviter des interférences. Ces problèmes de synchronisation apparaissent parce que l'intervalle entre deux moitiés d'image augmente (phases de construction de l'image). Mais ce système a également des inconvénients. Il était par exemple impossible de représenter sur l'écran des surfaces pleines au moyen du graphisme vectoriel.

> Le graphisme vectoriel s'est cependant maintenu jusqu'à nos jours car il se prête parfaitement bien à la représentation d'objets à trois dimensions. C'est pourquoi un chapitre entier de ce livre y sera également consacré.

Quelques années plus tard le moniteur de données put s'imposer grâce à l'accroissement continu de la capacité mémoire des ordinateurs. Ainsi apparut le premier système de graphisme informatique fondé sur une grille; la résolution était cependant encore très faible. A l'aide des caractères de l'écran les scientifiques purent pour la première fois sortir sur un appareil de visualisation de données des graphiques simples tels que des histogrammes rudimentaires ou tels que l'illustration grossière d'une courbe sinusoïdale. Le graphique obtenu pouvait aussi être transformé en n'importe quelle autre structure sans grand travail de programmation. On pouvait également mélanger sans problème du texte avec du graphisme puisque le graphisme lui-même n'était obtenu qu'à partir de caractères de texte.

Ces dessins rudimentaires ne suffirent naturellement pas à satisfaire à jamais les programmeurs; il était par exemple impossible de dessiner des cercles ou des droites sur l'écran. Pour rendre cela possible, on intégra dans le jeu de caractères des caractères spéciaux tels que les traits, les cercles et les triangles.

Les graphiques obtenus avec ces caractères étaient déjà nettement plus abstraits et permettaient de représenter des images plus complexes s'ils étaient habilement employés. Les PCs (=ordinateurs personnels) de la première génération n'avaient d'ailleurs pas d'autres possibilités graphiques que celles offertes par ce type de jeux de caractères. La résolution graphique du légendaire TRS 80 (128 fois 48 points) était par exemple basée sur 64 caractères spéciaux prédéfinis.

Mais cette méthode avait encore des inconvénients. D'abord, du fait de la faiblesse de la résolution, on ne pouvait représenter n'importe quelle forme et d'autre part une animation graphique sans à-coups restait impossible sur de nombreux ordinateurs. Pour pouvoir éliminer également ces deux facteurs, un jeu de caractères librement programmable fut intégré dans l'ordinateur. Il permettait au programmeur de définir des caractères personnels, adaptés à une application particulière, ce qui permettait, avec un travail de programmation non négligeable, de pouvoir manipuler n'importe quel point de l'écran graphique.

Nous bénéficions aujourd'hui, en tant que possesseurs du CPC AMSTRAD, du graphisme informatique fondé sur un système de grille. Cette méthode consiste à réserver une partie importante de la mémoire de travail pour le stockage d'une page de graphisme à haute résolution. Chaque bit de cette mémoire représente dans un tel système un point allumé ou éteint sur l'écran. Cette méthode permet de composer point par point des objets de toute sorte, comme dans une mosaïque. La résolution ainsi obtenue ne dépend ici que de la place mémoire disponible. Si on veut produire un graphisme en couleur, il faut même multiplier par un certain facteur la quantité de cellules de mémoire consacrée à la mémoire écran.

Vous retrouverez dans cet ouvrage toutes les méthodes de production de graphisme exposées ci-dessus car le CPC AMSTRAD dispose de propriétés graphiques exceptionnelles malgré son prix extraordinairement bas. Nous vous apprendrons à employer de façon optimale les différentes méthodes de façon à ce que vous puissiez à l'avenir exploiter pleinement les possibilités de votre ordinateur.

2. Point, virgule, trait...

En examinant le message initial après allumage des ordinateurs CPC, rien n'indique quelles propriétés graphiques insoupçonnées ces ordinateurs recèlent. Ils offrent en effet, par exemple, une résolution graphique identique (640 fois 200 points) à celle d'un IBM PC avec carte graphique couleur, et cela en série. Mais ce n'est pas tout, la résolution des CPCs n'est pas fixée une fois pour toutes mais peut au contraire être définie sur trois niveaux différents. Chacun de ces niveaux, également appelés modes, détermine outre la finesse de la résolution le nombre de couleurs disponibles.

Comme l'une des particularités de tous les CPCs est de permettre de représenter également du texte sur l'écran en mode graphique, les trois modes dont nous venons de parler agissent aussi bien sur la représentation du texte que sur celle du graphisme. La possibilité de représenter du texte en mode graphique fait que le CPC peut sans aucun problème mélanger à loisir les éléments de texte et de graphisme.

Les modes déterminent en premier lieu la finesse de résolution pouvant être obtenue sur l'écran mais aussi, comme le texte est représenté sous forme graphique, le nombre de caractères pouvant être représentés sur une ligne. Comme la place mémoire réservée aux informations représentées sur l'écran est limitée, il s'ensuit nécessairement qu'une résolution croissante entraîne une diminution du nombre de couleurs pouvant être représentées simultanément sur l'écran. Il en va nécessairement ainsi parce que les informations sur les couleurs sont liées de façon inséparable aux informations d'image et que ces deux sortes d'informations sont placées ensemble dans la mémoire écran. Vous voyez dès à présent toute l'importance des modes, importance qui nous a conduit à commencer par la description des modes notre exploration des possibilités graphiques du CPC.

Après allumage, l'ordinateur se trouve automatiquement en MODE 1, c'est-à-dire qu'il peut représenter 40 caractères par ligne avec quatre des 27 couleurs possibles. Pour passer alors dans un autre mode, il suffit de donner l'instruction MODE suivie d'un espace et du numéro du mode voulu.

Après confirmation de l'entrée avec la touche RETURN, l'ordinateur effacera l'écran, activera le nouveau mode et se déclarera prêt à recevoir la prochaine instruction avec un "Ready" dans le coin supérieur gauche de l'écran (position HOME = maison, origine).

La table suivante vous montre les effets des différentes instructions MODE disponibles :

Mode 0 :	20 caractères par ligne 160 fois 200 points 16 des 27 couleurs
Mode 1 :	40 caractères par ligne 320 fois 200 points 4 des 27 couleurs
Mode 2 :	80 caractères par ligne 640 fois 200 points 2 des 27 couleurs (monochrome)

Il est évident que le mode le plus intéressant variera en fonction des circonstances et de vos objectifs; nous conseillons par exemple le MODE 2 avec résolution de 80 caractères pour le traitement de textes et le MODE 1 pour les graphiques devant être rendus sur l'écran de façon assez fine. Reste le MODE 0 qui ne sera employé, du fait de sa faible résolution écran, que lorsqu'on voudra représenter en même temps le plus grand nombre de couleurs possible. Exemples d'applications: les programmes d'analyse et de jeux.

2.1 LE PREMIER POINT APPARAIT

Le graphisme du CPC AMSTRAD appartient, comme nous vous l'avons déjà indiqué dans le chapitre d'introduction, à la catégorie des systèmes graphiques basés sur une grille. La notion de "grille" signifie ici que l'image est composée à partir de nombreux petits points qui peuvent être allumés, éteints ou coloriés un par un. Ces points sont appelés en anglais "pixels" ce qui est une abréviation du terme "picture elements", autrement dit éléments d'image.

La gestion d'un graphisme de grille de ce type est très compliquée, le chemin suivi par un pixel pour apparaître sur l'écran est parfaitement incompréhensible pour un non-initié car cela suppose de solides connaissances de programmation en langage machine. Le CPC dispose donc heureusement d'un interpréteur BASIC très puissant, doté d'options graphiques exceptionnelles, ce qui fait que même un pur programmeur BASIC peut exploiter à fond une grande partie des possibilités graphiques de son ordinateur. Cela est possible même lorsqu'on ne sait pas précisément comment les choses fonctionnent à l'intérieur de l'ordinateur lui-même.

L'instruction PLOT est un bon exemple d'une instruction de ce type en BASIC Locomotive. Cette instruction permet en effet de fixer un point sur l'écran. Pour préciser l'action de l'instruction PLOT, quatre paramètres sont nécessaires. Les deux premiers paramètres fixent la position absolue dans laquelle le point doit être fixé; ces indications sont absolument indispensables pour que la syntaxe de l'instruction soit complète. Les deux paramètres suivants sont facultatifs. Ils permettent de déterminer la couleur du point d'image à fixer ainsi que son interaction avec le fond de l'écran. Ces dernières indications ne nous intéresseront cependant pas pour le moment.

Syntaxe :

PLOT coordonnée X, coordonnée Y [,crayon couleur [,mode couleur]]

Le mode couleur ne peut être indiqué sur le CPC 464.

Examinons d'abord l'instruction PLOT en détail : PLOT a pour premier effet de placer simplement un point sur l'écran; suivant le MODE (finesse de résolution) sélectionné, la sortie obtenue sur l'écran sera cependant différente. Le point fixé paraîtra en effet plus ou moins gros en fonction de MODE. Si on étudie ce phénomène de plus près, on constatera qu'un point en MODE 0 est exactement de la grosseur de deux points en MODE 1 ou de quatre en MODE 2. On s'apercevra en outre que le même point peut être fixé en MODE 0 avec huit instructions PLOT différentes et de même qu'un seul et même point peut être fixé en MODE 1 avec quatre instructions différentes et avec deux en MODE 2. L'explication en est très simple contrairement à ce qu'il peut sembler au premier abord.

Le CPC calcule, de façon interne, toujours d'après une résolution théorique de 640 fois 400 points. La résolution Y maximale correspond donc au double de la résolution Y effective. Ce fait est important pour éviter toute distorsion de l'image en MODE 2 et il explique le fait que chaque point d'image (indépendamment de MODE) puisse être adressé avec deux coordonnées Y différentes.

Comme la résolution interne de l'ordinateur est identique pour les trois modes dans le sens des X (c'est-à-dire horizontalement) à celle du MODE 2, cela signifie obligatoirement qu'un point apparaîtra plus gros dans un MODE de résolution inférieure à celle du MODE 2. C'est pourquoi un point se compose, arithmétiquement, en MODE 0 et en MODE 1 de plusieurs points et c'est pourquoi il peut de ce fait être adressé (c'est-à-dire défini dans l'espace) avec différentes coordonnées X. Le CPC convertit automatiquement les coordonnées entrées pour les adapter au niveau de résolution sélectionné et le point apparaît sur l'écran avec l'épaisseur imposée par le MODE fixé.

Cette méthode qui peut sembler un peu illogique au premier abord présente cependant un avantage décisif. L'utilisateur n'est pas en effet obligé de recalculer les coordonnées graphiques lorsqu'il sélectionne un mode différent. La figure 1, sur la page suivante, illustre encore une fois ce principe global :

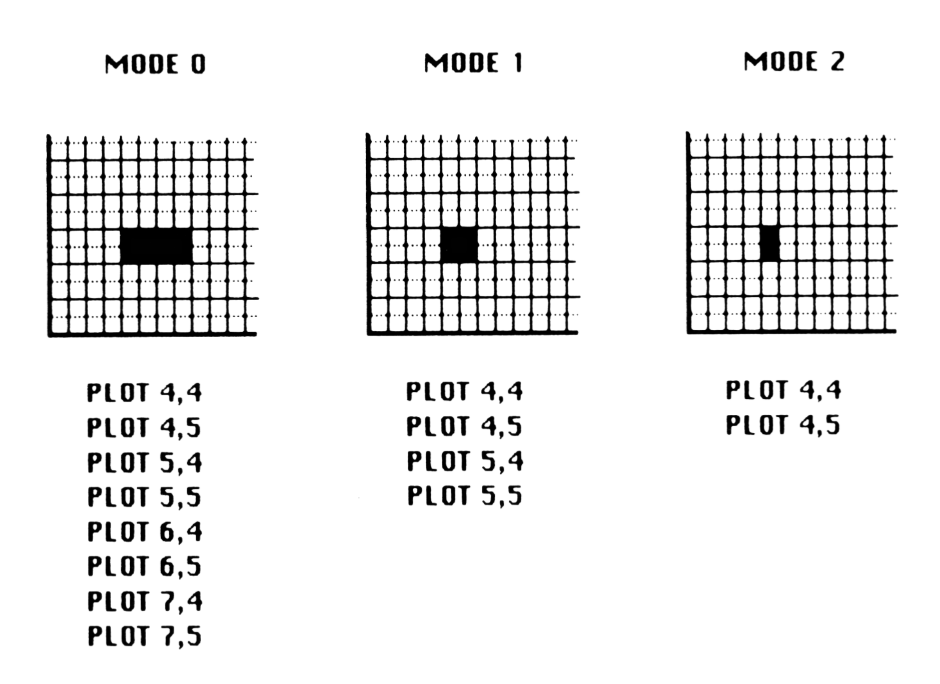


Figure 1 : **Effet des MODEs sur l'instruction PLOT**

Outre le principe dont nous venons de parler, l'instruction PLOT a encore une fonction dont l'importance ne doit pas être sous-estimée. En plus de la sortie directement apparente d'un point d'image, elle positionne également, par ses paramètres X/Y, un curseur graphique invisible.

Ce curseur graphique, contrairement au curseur de texte habituel, n'est jamais montré à l'écran. Il constitue simplement une position de référence gérée par l'ordinateur qui conserve les dernières coordonnées dans lesquelles une sortie sur l'écran graphique a été effectuée. La position du curseur graphique telle qu'elle a pu être par exemple modifiée par une instruction PLOT reste dans la mémoire de l'ordinateur même après exécution de cette instruction.

Ce fait est intéressant dans de nombreux cas car certaines instructions graphiques n'autorisent pas une définition intégrée de coordonnées, c'est-à-dire qu'elles ne déterminent les coordonnées qu'au vu de la position du curseur graphique. Pour que de telles instructions ne dépendent pas exclusivement des instructions avec définition de paramètres dont l'effet est visible sur l'écran, le jeu d'instructions de votre CPC comporte une instruction spéciale qui permet de positionner le curseur graphique sur n'importe quel point de l'écran, il s'agit de l'instruction MOVE. Sa syntaxe est identique à celle de l'instruction PLOT, à part bien sûr le mot-clé lui-même.

Syntaxe :

MOVE coordonnée X, coordonnée Y [,crayon couleur [,mode couleur]]

Le mode couleur ne peut pas être indiqué sur le CPC 464.

La figure 2, contrairement à la figure 1, matérialise simplement de façon symbolique la position du curseur graphique qui reste en réalité invisible sur l'écran. Elle ne signifie donc pas qu'un point soit mis ou ne soit pas mis à cet endroit. Les trois modes ont ici aussi les mêmes effets que pour PLOT.

L'instruction MOVE, c'est-à-dire le positionnement du curseur graphique, est surtout utilisée en liaison avec les instructions graphiques relatives ou avec la représentation de texte en mode graphique. Elle permet en effet de déterminer au point près le point de départ d'un texte. Les instructions graphiques relatives fonctionnent fondamentalement de la même façon que leurs équivalents absolus avec cette seule différence que leurs paramètres X et Y représentent une distance qui doit être ajoutée aux coordonnées du curseur graphique.

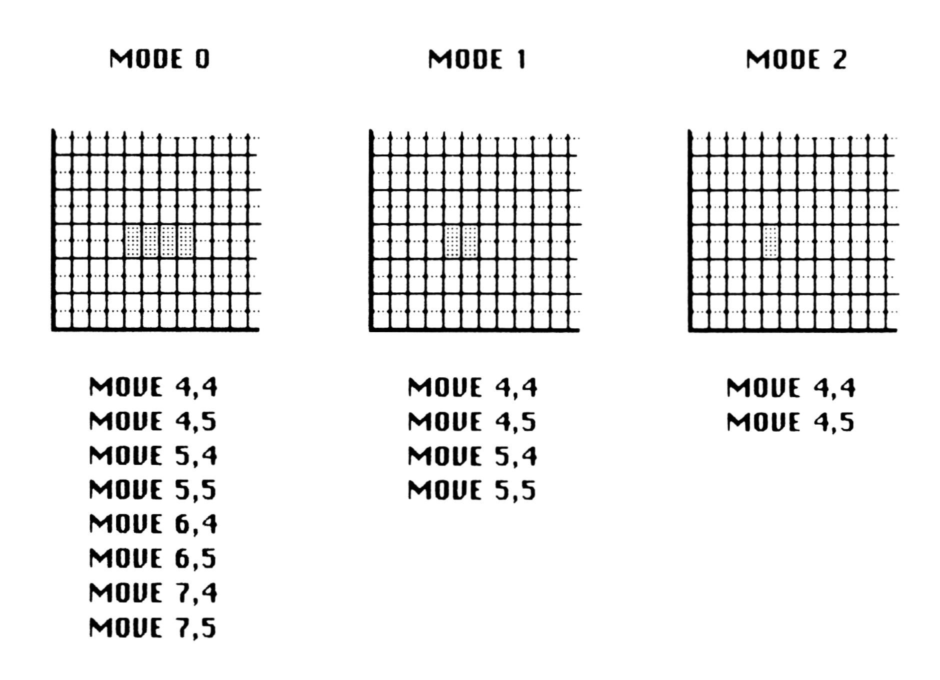


Figure 2 : **Effet des MODEs sur l'instruction MOVE**

2.2 DU POINT A LA LIGNE

Récapitulons un peu les outils dont nous avons appris à nous servir jusqu'ici. Nous savons que dans le cadre du graphisme de grille, toutes les sorties sur écran sont composées à la base de points isolés. Il doit donc être possible de créer un élément graphique plus grand, une ligne, en partant des seules instructions PLOT et MOVE.

De nombreux points allumés consécutivement sur l'écran donneront au spectateur l'impression qu'il s'agit d'une ligne continue. Avec une boucle FOR-TO-NEXT et l'instruction PLOT, on peut produire sur l'écran une ligne horizontale ou verticale. Les programmes suivants montrent comment une ligne peut être constituée à partir de nombreux points.

```
10 REM TRACER UNE LIGNE HORIZONTALE
20 CLS
30 FOR X=0 TO 639
40 PLOT X,200
50 NEXT
60 END
```

```
10 REM TRACER UNE LIGNE VERTICALE
20 CLS
30 FOR Y=0 TO 399
40 PLOT 320,Y
50 NEXT
60 END
```

Pendant l'exécution des programmes présentés ci-dessus, vous pouvez parfaitement suivre la fusion des différents points image. La vitesse d'exécution est très faible comme il s'agit ici de programmes BASIC et c'est pourquoi a été intégrée dans le système d'exploitation du CPC AMSTRAD une routine qui permet de dessiner des lignes beaucoup plus vite. Cette routine peut être appelée, à partir du niveau du BASIC, avec l'instruction DRAW.

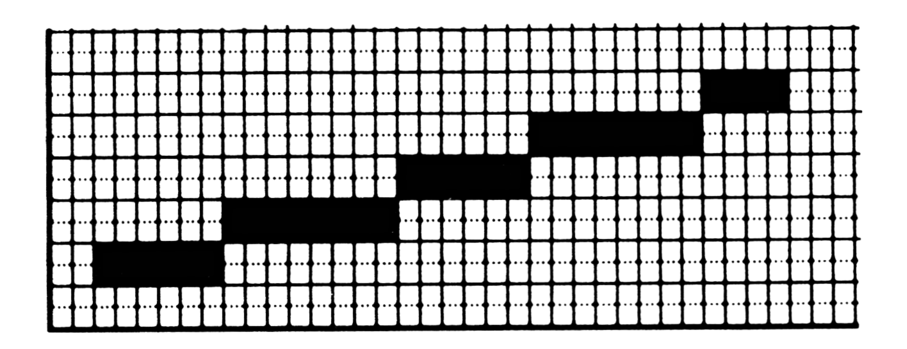
Syntaxe :

DRAW coordonnée X, coordonnée Y [,crayon couleur [,mode couleur]]

Le mode couleur ne peut pas être indiqué sur le CPC 464.

Les coordonnées de départ d'une ligne produite avec l'instruction DRAW correspondent à la position actuelle du curseur graphique dont nous avons déjà parlé. C'est pourquoi deux coordonnées suffisent à définir une instruction DRAW, ces deux coordonnées indiquant le but de la ligne à tracer. Ici aussi s'appliquent, comme pour toutes les autres instructions graphiques, les explications données pour PLOT et MOVE au sujet des différents modes graphiques. La figure 3 illustre à titre d'exemple la construction d'une ligne en MODE 1.

MODE 1



MOVE 2,2
MOVE 2,3
MOVE 3,2
MOVE 3,3

DRAW 34,12
DRAW 34,13
DRAW 35,12
DRAW 35,13

Figure 3 : **Exemple d'une ligne en MODE 1**

La ligne représentée peut être obtenue sur l'écran avec n'importe quelle combinaison d'une des instructions MOVE et d'une des instructions DRAW présentées dans la figure 3. Les coordonnées de départ et de destination peuvent d'ailleurs être librement interchangées car la routine DRAW du système d'exploitation replace automatiquement les coordonnées dans l'ordre qui convient.

2.3 FORMES GEOMETRIQUES SIMPLES

Certains interpréteurs BASIC commercialisés fournissent des instructions dessinant sur l'écran des structures géométriques de base simples telles que les triangles, les rectangles et les cercles. Le BASIC Locomotive ne comporte cependant pas de telles instructions, ce qui signifie pour l'utilisateur qu'il doit les reconstituer lui-même.

Il lui faut pour cela se servir de combinaisons des instructions MOVE, PLOT et DRAW. Les structures géométriques à bords pointus sont assez faciles à réaliser car il n'est pas nécessaire pour produire de tels objets d'avoir recours à des fonctions trigonométriques. Le programme suivant vous propose une possibilité de dessiner un carré sur l'écran avec des instructions DRAW. Vous pourrez en même temps constater une fois de plus tout l'intérêt de l'emploi du curseur graphique dont nous avons parlé plus haut car chaque instruction DRAW n'a besoin que de deux coordonnées de destination (X2, Y2) pour dessiner une ligne. Les coordonnées de départ de la ligne (X1, Y1) correspondent à la position du curseur graphique qui est automatiquement actualisée à la suite des coordonnées de destination indiquées pour chaque instruction DRAW.

```
10 REM DESSIN D'UN CARRE
20 MODE 1
30 CLS
40 MOVE 100,100
50 DRAW 100,200
60 DRAW 200,200
70 DRAW 200,100
80 DRAW 100,100
90 END
```

Avec la même technique que celle utilisée pour produire notre carré, nous pouvons naturellement réaliser n'importe quelles autres structures telles que les triangles ou les polygones. Il suffit de reprendre le programme ci-dessus en modifiant simplement le nombre et les paramètres des instructions DRAW. Pour produire un triangle, il nous faudra par exemple trois instructions DRAW.

```
10 REM DESSIN D'UN TRIANGLE
20 MODE 1
30 CLS
40 MOVE 100,100
50 DRAW 150,200
60 DRAW 200,100
70 DRAW 100,100
80 END
```

Les cercles et les ellipses ne peuvent par contre être calculés qu'à l'aide d'algorithmes compliqués car leur périmètre ne peut être calculé qu'avec les produits COS(ALPHA)*X et SIN(ALPHA)*Y qu'il faut ensuite amener sur l'écran avec l'instruction PLOT. La production d'un cercle prend par ailleurs beaucoup de temps car il faut calculer un sinus et un cosinus pour chaque point à dessiner. Le programme qui suit maintenant trace un cercle d'après cette méthode classique. A la suite de ce programme, nous découvrirons une technique qui permet de réduire considérablement le travail de calcul nécessaire.

```
10 CLS
20 DEG
30 FOR A=1 TO 360
40 MOVE 320,200
50 PLOT 320+190*COS(A),200+190*SIN(A)
60 NEXT
```

Un cercle peut être défini comme un polygone possédant un nombre illimité d'angles. Nous pouvons utiliser ce fait pour remplacer la routine de calcul ci-dessus par un algorithme qui dessinera en réalité un polygone dont les angles seront reliés par des lignes. Le nombre d'angles devra être choisi de façon à ce que le polygone produit ne puisse se distinguer, du fait de la résolution de l'écran, d'un cercle.

Pour augmenter encore la vitesse d'exécution de la routine, on peut utiliser la routine dite du "quart de cercle". Cette technique du quart de cercle consiste à ne calculer qu'un quart du cercle à dessiner et à construire les trois autres quarts de la circonférence du cercle par réflexion comme avec un miroir.

Le programme suivant réalise le dessin de polygone en combinaison avec la technique du quart de cercle que nous venons de vous présenter. Comme elle ramène au minimum nécessaire le temps de calcul d'une circonférence de cercle, nous l'utiliserons désormais dans tous les programmes BASIC fonctionnant avec une fonction de cercle.

```
100 CLS
110 '
120 DEG
130 '
140 INPUT "CENTRE COORDONNEE X";XP%
160 INPUT "       COORDONNEE Y";YP%
180 '
190 PRINT
200 '
210 INPUT "RAYON  PROFONDEUR X";XR%
230 INPUT "       PROFONDEUR Y";YR%
250 '
260 CLS
270 '
280 'PREMIERE COORDONNEE
290 '
300 j%=0
310 GOSUB 560
320 x1%=x2%
330 y1%=y2%
340 '
350 'AUTRES COORDONNEES
360 '
370 FOR j%=6 TO 90 STEP 6
380 GOSUB 560
390 '
400 'DESSIN DU CERCLE
410 '
420 MOVE xp%+x1%,yp%+y1%
430 DRAW  xp%+x2%,yp%+y2%
440 MOVE xp%+x1%,yp%-y1%+1
450 DRAW  xp%+x2%,yp%-y2%+1
460 MOVE xp%-x1%+1,yp%-y1%+1
470 DRAW  xp%-x2%+1,yp%-y2%+1
480 MOVE xp%-x1%+1,yp%+y1%
490 DRAW  xp%-x2%+1,yp%+y2%
500 '
510 x1%=x2%:y1%=y2%
520 '
530 NEXT
540 '
550 END
```

```
560 '
570 'DETERMINER LES COORDONNEES
580 '
590 x2%=INT(COS(j%)*xr%+0.5)
600 y2%=INT(SIN(j%)*yr%+0.5)
610 '
620 RETURN
```

Description du programme :

100-260 *Initialisation*
Cette partie du programme vide l'écran, fixe la mesure en degrés nécessaire pour le sinus et le cosinus et charge en lignes 140 et 150 les coordonnées du centre du cercle dans les variables XP% et YP% à l'aide de l'instruction INPUT. Les lignes 210 et 230 transmettent par ailleurs les longueurs X et Y du rayon aux variables XR% et YR% avec des instructions INPUT. Comme cette routine de cercle permet également de dessiner des ellipses, on demande maintenant à l'utilisateur d'entrer deux rayons. Si le rayon dans la direction X est le même que celui dans la direction Y, le résultat produit sur l'écran sera un cercle (forme particulière d'ellipse).

270-550 Comme le dessin du cercle ne se fait pas avec l'instruction BASIC PLOT mais avec l'instruction DRAW, il faut chaque fois calculer un point de départ et un point de destination pour la ligne à dessiner. Le point de départ pour la première ligne (0 degré) est calculé dans la partie PREMIERE COORDONNEE du programme, les coordonnées se trouvant finalement dans les variables X1% et Y1%. Dans la boucle FOR-TO-NEXT des lignes 370 à 530, toutes les autres coordonnées (de 6 à 90 degrés) sont ensuite calculées par étapes de 6 degrés. En ce qui concerne les coordonnées, remarquons encore qu'elles figurent toutes sans exception dans le premier quart du cercle.

Les coordonnées pour les trois autres quarts de cercle sont calculées par réflexion (par symétrie) par rapport au centre du cercle. En examinant la partie DESSIN DU CERCLE du programme, on comprend vite comment le dessin du cercle est précisément réalisé. Une fois une ligne tracée, son point final devient le point de départ de la ligne suivante (ligne 510). Une fois que le cercle a été entièrement dessiné, le programme rencontre END.

560-Fin — Le sous-programme DETERMINER LES COORDONNEES calcule les coordonnées des points placés sur la circonférence du premier quart de cercle, à l'aide des outils de la géométrie. J% contient l'indication de la position en degrés pour laquelle doivent être calculées les coordonnées correspondantes (X2%, Y2%). Après calcul d'un couple de coordonnées, on retourne au programme principal.

2.4 LA COULEUR ENTRE DANS LE JEU

Nous n'avons pu jusqu'ici réaliser que des dessins monochromes, c'est-à-dire en deux couleurs. Le BASIC AMSTRAD vous offre cependant des possibilités très étendues pour introduire de la couleur dans vos créations.

Comme nous l'indiquions déjà dans le chapitre introductif, notre CPC AMSTRAD dispose de 27 couleurs différentes. Suivant le MODE employé, 2, 4 ou 16 de ces couleurs peuvent être utilisées simultanément. Les 27 couleurs de base peuvent être comparées à une palette qui mettrait à notre disposition 27 peintures différentes. L'utilisateur disposera donc de 2, 4 ou 16 pinceaux auxquels il pourra chaque fois attribuer une couleur. C'est à cela que sert sur le CPC AMSTRAD l'instruction INK. Ink signifie encre en anglais. L'instruction Ink permet donc de colorier un pinceau avec une couleur déterminée tirée de notre palette. Sur le plan de la programmation, cela signifie que deux paramètres doivent être attribués à l'instruction INK. En voici un exemple :

INK 2,5

Le premier paramètre indique ici le pinceau utilisé. Sa valeur peut varier de 0 à 15 en fonction du MODE. Le second paramètre représente le code de la couleur de base souhaitée. Sa valeur doit toujours être comprise entre 0 et 26. Contrairement à ce qui se passe avec un "vrai" pinceau, il est toutefois également possible d'attribuer deux couleurs au pinceau. Ces deux couleurs ne seront pas mélangées mais elles seront sorties par intermittence pour le pinceau correspondant. Pour intégrer ce second paramètre de couleur dans la syntaxe de l'instruction INK, il suffit d'ajouter un nombre supplémentaire compris entre 0 et 26, après une virgule.

Une fois que toutes les couleurs de pinceau ont été ainsi définies, l'utilisateur n'a plus qu'à indiquer à son ordinateur quel pinceau il veut utiliser pour le dessin. Le BASIC Locomotive du CPC comporte à cet effet l'instruction PEN. L'instruction PEN doit toujours comprendre un paramètre qui indique le numéro du pinceau voulu. La valeur de ce paramètre peut être comprise, suivant le MODE activé, entre 0 et 1, 0 et 3 ou 0 et 15.

PEN 11

activera par exemple le pinceau 11, c'est-à-dire que les sorties sur écran se feront désormais dans la couleur du pinceau 11.

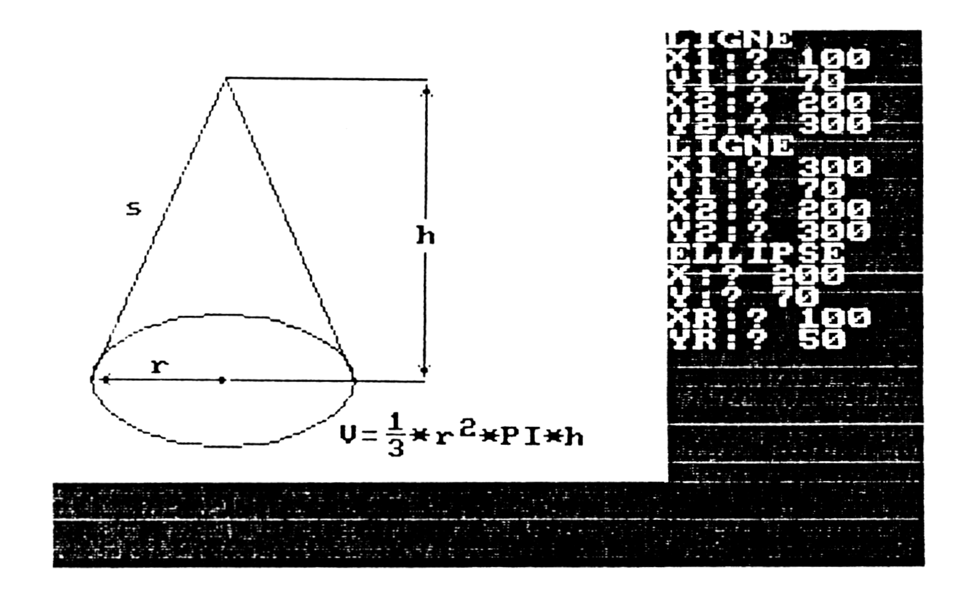
Notez par ailleurs que le manuel AMSTRAD ne parle pas de pinceau mais de crayon de couleur. Nous avons malgré cela préféré utiliser le terme de pinceau dans les exemples que nous venons d'expliquer pour ne pas compliquer encore la description de phénomènes qui ne sont déjà pas très simples. Il nous semble en effet que la comparaison avec un pinceau est plus appropriée puisqu'il est possible de changer la couleur d'un pinceau mais pas celle d'un crayon de couleur. Comme AMSTRAD utilise cependant le terme de "crayon de couleur" et non de "pinceau", nous vous demanderons désormais de vous en tenir au terme utilisé par AMSTRAD.

2.5 UN PETIT PROGRAMME DE DESSIN

Nous avons réuni, en conclusion de ce chapitre, toutes les méthodes étudiées jusqu'ici pour produire des dessins sur l'écran pour en faire un programme d'application complet, le "petit programme de dessin".

Le programme divise l'écran en trois zones ayant chacune une fonction tout à fait différente. La zone placée sur le bord gauche de l'écran représente la feuille de dessin sur laquelle vous réaliserez vos dessins. Sous la feuille de dessin se trouve une seconde zone dans laquelle sont énumérées les options et leurs méthodes de sélection rapide. Si on veut par exemple amener un point sur l'écran, on cherchera dans la liste des options l'option "(P)oint". On tapera ensuite sa sélection rapide (la lettre entre parenthèses), un "P" donc en l'occurrence. Sur la troisième fenêtre, qui se trouve sur le bord droit de l'écran, le message "POINT" apparaîtra. La troisième fenêtre est une fenêtre de dialogue qui affiche d'une part toutes les options appelées et qui sert d'autre part à demander les coordonnées nécessaires pour le dessin.

La copie d'écran 1 montre la division de l'écran dans le programme de dessin; on distingue nettement les différentes fonctions de la feuille de dessin, de la fenêtre d'option et de la fenêtre de dialogue.



Copie d'écran 1 : **Exemple d'application du programme de dessin**

Pour représenter un point, il faut entrer deux coordonnées qui déterminent la position du point sur la feuille de dessin. Si on veut placer sur la feuille de dessin d'autres éléments du programme de dessin, on peut utiliser les coordonnées écrites dans la fenêtre de dialogue comme points de départ.

Outre la possibilité d'amener des points sur la feuille de dessin, le programme de dessin offre aussi les options suivantes :

- **(L)igne**
  Trace une ligne de la position X1/Y1 à la position X2/Y2.

- **(C)arré**
  Dessine sur la feuille de dessin un carré dont l'angle inférieur gauche est déterminé par les coordonnées X1/Y1 et donc l'angle supérieur droit est déterminé par les coordonnées X2/Y2.

- **(E)llipse**
  Dessine sur la feuille de dessin un cercle ou une ellipse. Le centre de cette forme géométrique est fixé par les coordonnées X1 et Y1. Les indications XR et YR permettent de fixer le rayon dans les directions X et Y. Si les valeurs entrées pour XR et YR sont identiques, c'est un cercle (forme particulière d'ellipse) qui apparaîtra sur la feuille de dessin.

- **(T)exte**
  L'option "Texte" permet de sortir des caractères sur l'écran. Les entrées X et Y déterminent la position de l'angle supérieur gauche de la matrice (cadre théorique) dans laquelle est inscrit le premier caractère du texte à sortir.

- **(D)isque**
  L'option "Disque" permet de sauvegarder sur disquette un dessin que vous avez réalisé. Le dessin est stocké sous forme d'un fichier binaire qui pourra être ensuite ramené sur l'écran à tout moment, même en dehors du programme de dessin (LOAD "Nom.BIN").

- **(O)uvrir**
  Cette option permet de ramener sur la feuille de dessin un dessin stocké sur disquette sous forme de fichier binaire. Après chargement d'un dessin, toutes les indications de la fenêtre de dialogue sont effacées.

- **(F)in**
  Cette option permet de mettre fin au programme après avoir répondu à une question de sécurité. Une fois que vous êtes passé par cette partie du programme, tous les dessins non sauvegardés sont perdus irrémédiablement.

C'est en essayant de réaliser des dessins avec ce programme de dessin que vous apprendrez le mieux à vous en servir. Essayez de comprendre comment le programme fonctionne en lisant la description du programme au fur et à mesure que vous dessinerez.

```
100 MODE 1
110 '
120 CM$="PLCETDONF"
130 '
140 DEG
150 '
160 ORIGIN 1,64,1,479,399,64
170 '
180 BORDER 13
190 '
200 INK 0,26
210 INK 1,0
220 INK 2,13
230 INK 3,8
240 '
250 WINDOW #0,1,30,1,21
260 WINDOW #1,1,30,22,25
270 WINDOW #2,31,40,1,25
280 '
290 PAPER #0,0
300 PEN #0,1
310 CLS #0
320 '
330 PAPER #1,2
340 PEN #1,3
```

```
350 CLS #1
360 '
370 PAPER #2,2
380 PEN #2,0
390 CLS #2
400 '
410 'SORTIR LES OPTIONS
420 '
430 PRINT #1,"(P)OINT   (L)IGNE   (C)ARRE   (E)LLIPSE (T)EXTE
 (D)ISQUE  (O)UVRIR  (N)OUVEAU (F)IN"
440 '
450 'SAUT AUX DIFFERENTS SOUS-PROGRAMMES
460 '
470 IUV$=INKEY$
480 IUV$=UPPER$(IUV$)
490 '
500 FOR IX=1 TO 9
510 IF IUV$<>MID$(CM$,IX,1) THEN NEXT:GOTO 440
520 ON IX GOSUB 550,650,780,940,1420,1580,1670,1770,1890
530 '
540 GOTO 440
550 '
560 'FIXER UN POINT
570 '
580 PRINT #2,"POINT"
590 INPUT #2,"X:";XP
600 INPUT #2,"Y:";YP
610 '
620 PLOT XP,YP
630 '
640 RETURN
650 '
660 'TRACER UNE LIGNE
670 '
680 PRINT #2,"LIGNE"
690 INPUT #2,"X1:";XP1
700 INPUT #2,"Y1:";YP1
710 INPUT #2,"X2:";XP2
720 INPUT #2,"Y2:";YP2
730 '
740 MOVE XP1,YP1
750 DRAW XP2,YP2
```

```
760 '
770 RETURN
780 '
790 'DESSINER UN CARRE
800 '
810 PRINT #2,"CARRE"
820 INPUT #2,"X1:";XP1
830 INPUT #2,"Y1:";YP1
840 INPUT #2,"X2:";XP2
850 INPUT #2,"Y2:";YP2
860 '
870 MOVE XP1,YP1
880 DRAW XP1,YP2
890 DRAW XP2,YP2
900 DRAW XP2,YP1
910 DRAW XP1,YP1
920 '
930 RETURN
940 '
950 'DESSINER UNE ELLIPSE
960 '
970 PRINT #2,"ELLIPSE"
980 INPUT #2,"X:";XP
990 INPUT #2,"Y:";YP
1000 XP%=INT(XP)
1010 YP%=INT(YP)
1020 INPUT #2,"XR:";XR
1030 INPUT #2,"YR:";YR
1040 XR%=INT(XR)
1050 YR%=INT(YR)
1060 '
1070 'PREMIERE COORDONNEE
1080 '
1090 j%=0
1100 GOSUB 1350
1110 x1%=x2%
1120 y1%=y2%
1130 '
1140 'AUTRES COORDONNEES
1150 '
1160 FOR j%=6 TO 90 STEP 6
1170 GOSUB 1350
```

```
1180 '
1190 'DESSINER UNE ELLIPSE
1200 '
1210 MOVE xp%+x1%,yp%+y1%
1220 DRAW xp%+x2%,yp%+y2%
1230 MOVE xp%+x1%,yp%-y1%+1
1240 DRAW xp%+x2%,yp%-y2%+1
1250 MOVE xp%-x1%+1,yp%-y1%+1
1260 DRAW xp%-x2%+1,yp%-y2%+1
1270 MOVE xp%-x1%+1,yp%+y1%
1280 DRAW xp%-x2%+1,yp%+y2%
1290 '
1300 x1%=x2%:y1%=y2%
1310 '
1320 NEXT
1330 '
1340 RETURN
1350 '
1360 'CALCUL DES COORDONNEES
1370 '
1380 x2%=INT(COS(j%)*xr%+0.5)
1390 y2%=INT(SIN(j%)*yr%+0.5)
1400 '
1410 RETURN
1420 '
1430 'SORTIR UN TEXTE
1440 '
1450 PRINT #2,"TEXTE"
1460 INPUT #2,"X:";XP
1470 INPUT #2,"Y:";YP
1480 INPUT #2,"TEXTE:";TE$
1490 '
1500 TAG
1510 '
1520 MOVE XP,YP
1530 PRINT TE$;
1540 '
1550 TAGOFF
1560 '
1570 RETURN
1580 '
1590 'SAUVEGARDER LA FEUILLE DE TRAVAIL
```

```
1600 '
1610 PRINT #2,"DISQUE"
1620 INPUT #2,"NOM:";NM$
1630 CLS #2
1640 SAVE NM$,B,&C000,&4000
1650 '
1660 RETURN
1670 '
1680 'CHARGER LA FEUILLE DE TRAVAIL
1690 '
1700 PRINT #2,"OUVRIR"
1710 INPUT #2,"NOM:";NM$
1720 IF UPPER$(RIGHT$(NM$,4))<>".BIN" THEN NM$=NM$+".BIN"
1730 CLS #2
1740 LOAD NM$
1750 '
1760 RETURN
1770 '
1780 'EFFACER LA FEUILLE DE TRAVAIL
1790 '
1800 PRINT #2,"NOUVEAU"
1810 PRINT #2,"SUR? ";
1820 S$=INKEY$
1830 IF S$="" THEN 1820
1840 IF S$<>"O" AND S$<>"o" THEN PRINT #2,"N":RETURN
1850 CLS #0
1860 CLS #2
1870 '
1880 RETURN
1890 '
1900 'FIN DU PROGRAMME
1910 '
1920 CALL &BB4E
1930 CALL &BBFF
1940 '
1950 END
```

Description du programme

100-390 Définitions et dimensionnements. La variable de texte CM$ se voit affecter une suite de caractères contenant les initiales des options du programme de dessin. A l'aide de CM$, la section de programme SAUT AUX DIFFERENTS SOUS-PROGRAMMES peut voir si la touche enfoncée doit ou non entraîner l'appel d'un sous-programme.

L'ordinateur passe en mesure en degrés, la couleur du cadre est fixée (BORDER) et les couleurs des crayons de couleur 0 à 3 sont définies. Enfin trois fenêtres sont constituées qui auront la signification suivante :

#0 : Feuille de travail sur laquelle le dessin sera réalisé. Les limites de cette fenêtre coïncident avec la délimitation de la zone graphique de l'écran qui a été définie en ligne 160.

#1 : C'est dans cette fenêtre que sont sorties les options du programme (ligne 430) où elles restent pendant tout le déroulement du programme.

#2 : Fenêtre de dialogue. C'est à travers cette fenêtre qu'est conduit le dialogue entre l'ordinateur et l'utilisateur. L'ordinateur indique par exemple ici dans quelle sous-routine il se trouve ou bien il vous demande d'indiquer des coordonnées. L'utilisateur tape alors au clavier les indications demandées qui sont ensuite copiées dans la fenêtre #2.

Une fois les trois fenêtres définies, chacune se voit attribuer des couleurs de premier plan et de fond propres et leur contenu est effacé. Elles sont alors prêtes pour les sorties du programme.

400-430 *Sortir les options*
La partie du programme SORTIR LES OPTIONS écrit dans la fenêtre #1 la liste des possibilités offertes par le programme et vous indique à travers quelle touche chaque option peut être appelée. La fenêtre #1 ne sert pas à autre chose qu'à sortir ces informations qui y demeureront pendant tout le déroulement du programme.

440-540 *Saut aux différents sous-programmes*
Ces lignes de programme constituent la boucle principale, c'est-à-dire en quelque sorte le poste d'aiguillage du programme, d'où partent tous les sauts aux différents sous-programmes. C'est pour cela que l'entrée effectuée au clavier est comparée avec les différents éléments de la variable CM$ (ligne 510). Si aucune identité n'apparaît, la boucle Saut aux différents sous-programmes est parcourue une nouvelle fois. Si l'entrée correspond à un élément de CM$, on saute au sous-programme correspondant à l'entrée de l'utilisateur avec ON IX GOSUB, d'après la valeur actuelle de l'index IX. Après qu'on soit revenu du sous-programme appelé, la boucle principale est à nouveau traitée.

550-640 *Fixer un point*
Le fait d'appuyer sur la touche "P" déclenche un saut de la boucle principale à ce sous-programme. La fenêtre de dialogue (#2) indique qu'il s'agit du sous-programme FIXER "POINT" et attend que vous entriez les coordonnées X et Y du point. Les valeurs à entrer doivent être comprises relativement à l'origine des coordonnées fixée (ORIGIN). Le point est dessiné sur la feuille de travail avec les coordonnées entrées (#0). Enfin retour à la boucle principale.

650-770 *Dessiner une ligne*
Si la touche "L" a été actionnée, la boucle principale saute à ce sous-programme. Sur la fenêtre de dialogue (#2), on vous indique qu'il s'agit du sous-programme DESSINER "LIGNE".

On attend alors dans la fenêtre de dialogue que vous entriez les coordonnées des point de départ (X1, Y1) et point final (X2, Y2). Conformément aux coordonnées entrées (relativement à l'origine fixée), le curseur graphique est amené, sur la feuille de travail, sur le point de départ de la ligne et la ligne est dessinée jusqu'au point final fixé. Retour à la boucle principale.

780-930 *Dessin d'un carré*
Après que la touche "C" ait été enfoncée, la boucle principale a sauté à ce sous-programme. Après l'indication "CARRE", la fenêtre de dialogue vous demande d'entrer quatre valeurs. Les deux premières valeurs (X1, Y1) représentent le coin inférieur gauche du carré. Le second couple de valeurs (X2, Y2) représente le coin supérieur droit du carré. Ces indications doivent elles aussi être comprises relativement à l'origine de coordonnées fixée. Le carré est dessiné sur la feuille de travail conformément aux valeurs entrées (lignes 870-910).

940-1410 *Dessin d'une ellipse*
Si vous appuyez sur la touche "E", la boucle principale saute à la ligne 940. Dans la fenêtre de dialogue apparaît l'indication DESSINER "ELLIPSE" sur le sous-programme actuellement traité. On vous demande alors les coordonnées X et Y du centre du cercle. Ces coordonnées sont immédiatement placées dans une variable entière (plus grande vitesse de traitement possible). Comme cette routine de cercle permet plus généralement de dessiner toutes sortes d'ellipses, on vous demande d'entrer deux rayons. Si le rayon dans la direction X est identique au rayon dans la direction Y, c'est un cercle (forme particulière d'ellipse) qui apparaîtra sur l'écran. Les deux rayons seront également stockés dans des variables entières (XR%, YR%).

Comme le dessin du cercle ne travaille pas avec l'instruction BASIC PLOT mais utilise l'instruction DRAW, il faut calculer chaque fois un point de départ

et un point final pour la ligne à dessiner. Le point de départ pour la première ligne (degré 0) est calculé dans la partie du programme PREMIERE COORDONNEE, les coordonnées se trouvent finalement dans les variables X1% et Y1%. Dans la boucle FOR-TO-NEXT des lignes 1160 à 1320 sont ensuite calculées toutes les autres coordonnées, par étape de 6 degrés (de 6 à 90 degrés). Remarquons encore, en ce qui concerne les coordonnées, qu'elles se trouvent toutes sans exception dans le premier quart de cercle. Les coordonnées pour les trois autres quarts de cercle sont produites par réflexion par rapport au centre du cercle. En examinant la partie du programme DESSINER CERCLE, vous comprendrez vite comment s'effectue concrètement le dessin d'un cercle. Une fois qu'une ligne a été tracée, son point final devient le point de départ pour la prochaine ligne (ligne 1300). Une fois que le cercle a été entièrement dessiné, retour à la boucle principale.

Le sous-programme CALCULER LES COORDONNEES n'est pas géré par la boucle principale mais est utilisé exclusivement par DESSINER CERCLE. C'est dans ce sous-programme que les coordonnées des points de la circonférence du premier quart de cercle peuvent être calculés à l'aide des outils de la géométrie. J% contient une indication en degrés à partir de la quelle seront calculées les coordonnées correspondantes (X2%, Y2%).

1420-1570 *Sortir un texte*

Si la touche "T" a été actionnée, la boucle principale saute ici. Sur la fenêtre de dialogue il vous est indiqué qu'il s'agit du sous-programme SORTIR "TEXTE". On vous demande ensuite d'indiquer la position du curseur graphique à partir de laquelle devra se faire la sortie de texte sur la feuille de travail. A la suite du message "TEXTE:", vous devez taper le texte qui devra apparaître sur la feuille de travail.

En lignes 1500 à 1550, la sortie de texte est reliée à la position du curseur graphique (TAG), le curseur graphique est ramené sur l'emplacement du dernier point entré, le texte (TE$) est sorti sur la feuille de travail et la sortie de texte est à nouveau reliée au curseur de texte (TAGOFF). Retour à la boucle principale.

1580-1660 *Sauvegarde de la feuille de travail*
Pour que le dessin que vous avez réalisé ne soit pas définitivement perdu dès que vous éteignez l'ordinateur, l'option SAUVEGARDER FEUILLE DE TRAVAIL a été intégrée dans le programme. Elle est obtenue en frappant la touche "D". Dans la fenêtre de dialogue apparaît l'indication "DISQUE" et on attend que vous entriez un nom de fichier sous lequel l'écran actuel sera sauvé sur disquette. Avant la sauvegarde de l'écran, la fenêtre de dialogue est effacée (CLS #2) pour éviter toute complication dans l'option CHARGER FEUILLE DE TRAVAIL. En ligne 1640, la mémoire écran est alors entièrement sauvegardée sur disquette sous forme d'un fichier binaire.

1670-1760 *Charger la feuille de travail*
Pour pouvoir à nouveau travailler sur un dessin transféré sur disquette avec l'option DISQUE, ce dessin doit être ramené dans la mémoire écran avec le sous-programme CHARGER FEUILLE DE TRAVAIL. C'est en appuyant sur la touche "O" pour OUVRIR que vous demandez à la boucle principale de sauter à cet endroit du programme. Dans la fenêtre de dialogue (#2) apparaît un message indiquant qu'il s'agit du sous-programme OUVRIR. Le nom de fichier entré, NM$, se voit ajouter l'extension ".BIN" pour fichier binaire s'il ne la contient pas encore. Après que la fenêtre de dialogue ait été effacée, le fichier sélectionné est chargé dans la mémoire écran et vous pouvez continuer à y travailler.

1770-1880 *Effacer la feuille de travail*

En appuyant sur la touche "N", vous faites sauter la boucle principale à la sous-routine EFFACER FEUILLE DE TRAVAIL. Sur la fenêtre de dialogue apparaît l'indication "NOUVEAU" et le programme s'assure, en vous demandant "SUR ?" que la touche "N" n'a pas été frappée par erreur. Si vous répondez autre chose que "O" ou "o" à la question "SUR ?", un "N" apparaît comme réponse dans la fenêtre de dialogue et le sous-programme rencontre un RETURN. Si vous avez confirmé "SUR ?", la feuille de travail et la fenêtre de dialogue sont effacées.

1890-Fin *Fin du programme*

En sautant aux vecteurs &BB4E TXT INITIALISE (initialisation complète de la section Texte) et &BBFF SCR INITIALISE (initialisation complète de la section écran), la sortie sur écran est ramenée sur sa valeur défaut, c'est-à-dire que les couleurs qui ont été redéfinies sont rétablies et que les fenêtres définies sont annulées.

3. Le jeu de caractères - une réserve d'éléments graphiques

Le jeu de caractères est certainement, sur chaque ordinateur, un des éléments les plus importants. C'est sur la base de ce jeu de caractères que se déroulent toutes les communications, c'est-à-dire bien sûr la communication entre l'utilisateur et l'ordinateur mais aussi par exemple l'échange de données entre ordinateurs.

Toute forme de communication est soumise depuis toujours à des tendances puissantes d'unification. Sans normalisation, il n'est pas garanti que deux partenaires voulant communiquer entre eux puissent finalement se comprendre. La réussite de la communication suppose donc que tous les participants utilisent le même code, c'est-à-dire un même système de caractères bien définis, pour se faire comprendre. Cela signifie aussi bien que deux hommes doivent parler la même langue pour se comprendre ou encore que deux ordinateurs doivent utiliser la même norme de caractères pour échanger des informations.

Pour permettre la communication entre systèmes informatiques différents, on a développé dans le domaine de l'informatique un code appelé code ASCII (American Standard Code for Information Interchange) qui constitue une norme en matière de jeux de caractères. Les constructeurs de systèmes électroniques ne sont bien sûr pas obligés d'utiliser ce standard mais il est néammoins respecté pratiquement sans exception parce qu'il rend possible la compatibilité entre les systèmes. Pour celui qui se contente d'être un utilisateur d'installations informatiques, le code utilisé par l'ordinateur est sans importance mais, par contre, tous ceux qui s'intéressent à la programmation des ordinateurs ne peuvent se dispenser d'étudier le code ASCII.

Le code ASCII établit une norme pour tous les caractères dont le numéro de code appartient à la zone pouvant être représentée au moyen de sept bits (0-127). Ce standard fait donc que les caractères dont les codes se situent en dessous de cette limite seront toujours les mêmes sur tous les ordinateurs se conformant à cette norme, comme c'est le cas notamment du CPC.

Ainsi est donc remplie la principale condition pour un échange de données fructueux entre deux ordinateurs différents.

La zone des 128 premiers caractères contient les minuscules et les majuscules, les nombres et les caractères spéciaux ainsi que certains caractères de commande qui jouent un rôle pour la communication entre deux ordinateurs. La table 1 vous indique où les caractères figurent dans le standard ASCII. Cette table représente en effet tous les caractères ASCII sans exception avec leur numéro de code en formes binaire, décimale et hexadécimale.

Examinons donc une case de la table ASCII. Légèrement à droite du milieu de la table, vous trouverez une case comportant un "A" et le nombre 65. Cette table vous indique donc que la lettre "A" est liée d'après la norme ASCII au numéro de code 65. Pour de nombreuses applications du code ASCII il sera intéressant de disposer non seulement du numéro de code décimal mais aussi de ses équivalents hexadécimal et binaire. Pour vous éviter un travail pénible de conversion d'un système numérique à l'autre, chaque code vous est également donné dans la table dans les deux autres systèmes numériques.

Revenons à l'exemple du "A"; si vous suivez la colonne dans laquelle figure la case marquée 65 vers le haut jusqu'au trait un peu plus épais, vous rencontrerez le nombre 4 qui est le premier chiffre du code exprimé en hexadécimal. Le second chiffre peut être trouvé en suivant la ligne dans laquelle figure le 65 vers la gauche jusqu'au trait un peu plus épais. Vous obtenez donc pour le "A" le numéro de code hexadécimal 41.

De la même façon, vous pouvez lire d'après la table le numéro de code binaire; il faut simplement, dans ce cas, que vous poursuiviez les colonnes et les lignes jusqu'au bord extérieur de la table. Le numéro de code binaire est toujours composé de 7 bits.

ASCII 7 Bits
(American Standard Code for Information Interchange)

N° de bit		000	001	010	011	100	101	110	111
654 / 3210	Code Hexa	0	1	2	3	4	5	6	7
0000	0	NUL *00*	DLE *16*	SP *32*	**0** *48*	@ *64*	**P** *80*	` *96*	**p** *112*
0001	1	SOH *01*	DC1 *17*	! *33*	**1** *49*	**A** *65*	**Q** *81*	**a** *97*	**q** *113*
0010	2	STX *02*	DC2 *18*	" *34*	**2** *50*	**B** *66*	**R** *82*	**b** *98*	**r** *114*
0011	3	ETX *03*	DC3 *19*	# *35*	**3** *51*	**C** *67*	**S** *83*	**c** *99*	**s** *115*
0100	4	EOT *04*	DC4 *20*	$ *36*	**4** *52*	**D** *68*	**T** *84*	**d** *100*	**t** *116*
0101	5	ENQ *05*	NAK *21*	% *37*	**5** *53*	**E** *69*	**U** *85*	**e** *101*	**u** *117*
0110	6	ACK *06*	SYN *22*	& *38*	**6** *54*	**F** *70*	**V** *86*	**f** *102*	**v** *118*
0111	7	BEL *07*	ETB *23*	' *39*	**7** *55*	**G** *71*	**W** *87*	**g** *103*	**w** *119*
1000	8	BS *08*	CAN *24*	(*40*	**8** *56*	**H** *72*	**X** *88*	**h** *104*	**x** *120*
1001	9	HT *09*	EM *25*	) *41*	**9** *57*	**I** *73*	**Y** *89*	**i** *105*	**y** *121*
1010	A	LF *10*	SUB *26*	• *42*	: *58*	**J** *74*	**Z** *90*	**j** *106*	**z** *122*
1011	B	VT *11*	ESC *27*	+ *43*	; *59*	**K** *75*	[*91*	**k** *107*	{ *123*
1100	C	FF *12*	FS *28*	, *44*	< *60*	**L** *76*	*92*	**l** *108*	| *124*
1101	D	CR *13*	GS *29*	– *45*	= *61*	**M** *77*	] *93*	**m** *109*	} *125*
1110	E	SO *14*	RS *30*	. *46*	> *62*	**N** *78*	*94*	**n** *110*	~ *126*
1111	F	SI *15*	US *31*	/ *47*	? *63*	**O** *79*	— *95*	**o** *111*	**DEL** *127*

Caractère de commande

Table 1 : **Le code ASCII 7 bits**

Comme c'est le cas pour tout autre type de norme, le standard du code ASCII est lui aussi sapé. C'est ainsi qu'il y a des ordinateurs qui suivent largement le standard ASCII mais en s'écartant malgré tout de la norme pour quelques caractères. Comme certains caractères sont sans importance pour les communications, les développeurs de systèmes les modifient volontiers. C'est d'ailleurs également le cas du CPC bien qu'avec un jeu de caractères de 256 caractères il y ait suffisamment de place dans les 128 caractères les plus élevés pour faire preuve de créativité.

Les 128 codes de caractères les plus élevés parmi les 256 mis à votre disposition par le système d'exploitation ne sont pas concernés par la norme ASCII 7 bits. Ils peuvent donc se voir attribuer n'importe quels caractères.

Comme c'est également le cas sur beaucoup d'autres ordinateurs dotés d'un jeu de caractères de 256 caractères, le bit de valeur 2^7 est destiné sur le CPC à vous ouvrir la voie des caractères graphiques. Certains efforts sont bien sûr faits pour introduire un code ASCII 8 bits qui normaliserait aussi cette zone du jeu de caractères mais ils sont restés vains jusqu'à ce jour.

Nous découvrirons au cours de ce chapitre quelles entorses au principe de l'"American Standard Code for Information Interchange" les développeurs du CPC ont faites et quels caractères supplémentaires ils ont ajoutés dans ses ROMs.

3.1 LE JEU DE CARACTERES DU CPC

Nous avons parlé des caractères et des numéros de code qui constituent la base de toute communication. Ces deux notions décrivent des choses qui restent cachées à l'intérieur de l'ordinateur. La communication entre deux ordinateurs peut se faire par transmission de numéros de code mais lorsque l'ordinateur sort sur l'écran un message destiné à son utilisateur, il ne le fait pas en se servant de numéros de code mais bien avec les caractères correspondants.

Pour que le CPC puisse sortir sur l'écran les codes de caractères produits sous forme de caractères, il possède dans son système d'exploitation des routines qui constituent ce qu'on appelle le générateur de caractères. Ces routines convertissent les codes de caractères en caractères pouvant être représentés sur l'écran. Ils emploient pour cela des modèles de bits (matrices de caractères) placés dans la ROM qui définissent la forme de chaque caractère. La taille des caractères sur l'écran est déterminée par le générateur de caractères en fonction du MODE fixé, de sorte qu'il est possible de faire représenter soit 20, soit 40 ou encore 80 caractères par ligne. Le générateur de caractères est également chargé de colorier les caractères pour la sortie sur l'écran en fonction des combinaisons de couleurs fixées.

Chaque matrice de caractère du CPC se compose de huit fois huit points qui sont placés dans la ROM du CPC sous forme binaire. Les matrices de caractère des 256 caractères disponibles occupent dans la ROM les adresses &3800 à &3FFF.

Si vous essayiez toutefois de lire les matrices de caractère avec la fonction BASIC PEEK, vous ne pourriez qu'échouer dans votre tentative. L'électronique du CPC n'autorise en effet aucun accès à la ROM à partir du niveau du BASIC. Il n'est donc pas possible de lire les matrices de caractère placées dans la ROM de cette façon. Si vous voulez malgré tout utiliser les matrices de caractères, vous ne pouvez le faire que si vous maîtrisez bien les mécanismes, qui ne peuvent être mis en oeuvre qu'en langage machine, qui permettent de commuter entre différentes configurations de la mémoire (RAM-ROM).

Le programme suivant écrit dans la mémoire du CPC une petite routine machine qui permet de lire la ROM et qui vous montre comment les matrices de caractère peuvent être lues. Le programme envoie sur l'écran tous les octets des matrices de caractères sous formes de valeurs hexadécimales.

```
100 'COMMUTATION RAM-ROM
110 '
120 DATA &01,&82,&7f          :'LD   BC,&7F82[1]
130 DATA &ed,&49              :'OUT  (C),C
140 DATA &1a                  :'LD   A,(DE)
150 DATA &32,&7f,&ab          :'LD   (&AB7F),A
160 DATA &c9                  :'RET
170 '
180 MEMORY &A000
190 '
200 FOR a=&AB70 TO &AB79
210 READ d
220 POKE a,d
230 NEXT a
240 '
250 'LIRE LE JEU DE CARACTERES
260 '
270 FOR A=&3800 TO &3FFF
280 CALL &AB70,A
290 PRINT HEX$(PEEK(&AB7F))
300 NEXT A
```

1) Notez que sur le CPC 464 les commentaires placés dans des instructions DATA provoquent des Syntax Error. Sur ce modèle, les commentaires ne doivent donc pas être entrés.

Les instructions machine que renferment les instructions DATA sont écrites dans la mémoire par la première boucle FOR-TO-NEXT et elles y forment une petite routine qui permet de lire la ROM du CPC. La seconde boucle utilise cette routine qu'il s'agit de lire un octet d'une matrice de caractère. Lors de l'appel de la routine machine (CALL) l'adresse à partir de laquelle doit être lue la ROM lui est transmise. L'octet que la routine trouve dans la cellule de mémoire concernée de la ROM est écrit par celle-ci dans l'adresse &AB7F de la mémoire RAM. Il pourra y être lu avec la fonction PEEK (ligne 290). La présentation de tous les caractères du CPC AMSTRAD que vous trouverez à la suite de quelques phrases d'explication est basée sur les résultats obtenus avec ce programme.

Le jeu de caractères du CPC AMSTRAD se compose de 256 caractères dont les matrices de caractère sont chacune composées de huit octets. Les matrices de caractère occupent donc 2 K octets de la mémoire ROM du CPC. Pour que vous puissiez vous faire une idée des 256 caractères de votre ordinateur et de leurs formes respectives, nous avons regroupé les octets obtenus avec le programme que nous vous avons présenté par groupes de huit, chaque groupe représentant la matrice d'un caractère. Comme la représentation hexadécimale fournie par le programme se prête parfaitement à une transformation mais pas à une représentation visuelle des caractères, nous avons converti les différents octets en leurs équivalents binaires. Les bits mis (=1) qui représentent des points marqués lors de la représentation d'un caractère ont été figurés par un "■" alors que les bits annulés sont figurés par un espace. C'est ainsi qu'ont été reconstituées les matrices huit sur huit de chaque caractère sous une forme qui correspond à la représentation des caractères sur l'écran.

La représentation du jeu de caractères contient cependant encore d'autres informations que la matrice de chaque caractère en représentations binaire et hexadécimale. Il s'agit en effet des adresses de la ROM où peuvent être trouvés les différents octets de la matrice. Les adresses sont imprimées en forme hexadécimale. Vous pouvez les utiliser dans le programme de lecture des matrices de caractère imprimé plus haut pour examiner une matrice de caractère isolément.

La dernière information que vous puissiez tirer de cette présentation est le code de caractère figurant au dessus de chaque matrice de caractère. Utilisé dans une instruction PRINT CHR$, ce code vous permet d'envoyer sur l'écran le caractère correspondant.

Si en examinant le jeu de caractères et sa représentation vous accordez une attention particulière aux caractères 32 à 127, vous y retrouverez les principes de la norme ASCII tels que nous les avons décrits. Nous vous demandons de ne pas étudier de trop près pour le moment les caractères de commande ASCII qui figurent avant cette zone car un chapitre particulier (3.2) traitera en détail des caractères de commande.

```
        0                    1                    2                    3

3800 ■■■■■■■■ FF    3808 ■■■■■■■■ FF    3810    ■■    18    3818       ■■ 03
3801 ■■    ■■ C3    3809 ■■       C0    3811    ■■    18    3819       ■■ 03
3802 ■■    ■■ C3    380A ■■       C0    3812    ■■    18    381A       ■■ 03
3803 ■■    ■■ C3    380B ■■       C0    3813    ■■    18    381B       ■■ 03
3804 ■■    ■■ C3    380C ■■       C0    3814    ■■    18    381C       ■■ 03
3805 ■■    ■■ C3    380D ■■       C0    3815    ■■    18    381D       ■■ 03
3806 ■■    ■■ C3    380E ■■       C0    3816    ■■    18    381E       ■■ 03
3807 ■■■■■■■■ FF    380F ■■       C0    3817 ■■■■■■■■ FF    381F ■■■■■■■■ FF

        4                    5                    6                    7

3820     ■■   0C    3828 ■■■■■■■■ FF    3830          00    3838   ■■■■   3C
3821    ■■    18    3829 ■■    ■■ C3    3831        ■ 01    3839  ■■  ■■  66
3822   ■■     30    382A ■■■  ■■■ E7    3832       ■■ 03    383A ■■    ■■ C3
3823  ■■■■■■  7E    382B ■■ ■■ ■■ DB    3833      ■■  06    383B ■■    ■■ C3
3824     ■■   0C    382C ■■ ■■ ■■ DB    3834 ■■  ■■   CC    383C ■■■■■■■■ FF
3825    ■■    18    382D ■■■  ■■■ E7    3835  ■■■■    78    383D   ■  ■   24
3826   ■■     30    382E ■■    ■■ C3    3836   ■■     30    383E ■■■  ■■■ E7
3827          00    382F ■■■■■■■■ FF    3837          00    383F          00
```

8		9		10		11	
3840	00	3848	00	3850	18	3858	18
3841	00	3849	00	3851	18	3859	3C
3842	30	384A	0C	3852	18	385A	7E
3843	60	384B	06	3853	18	385B	DB
3844	FF	384C	FF	3854	DB	385C	18
3845	60	384D	06	3855	7E	385D	18
3846	30	384E	0C	3856	3C	385E	18
3847	00	384F	00	3857	18	385F	18

12		13		14		15	
3860	18	3868	00	3870	3C	3878	3C
3861	5A	3869	03	3871	66	3879	66
3862	3C	386A	33	3872	FF	387A	C3
3863	99	386B	63	3873	DB	387B	DB
3864	DB	386C	FE	3874	DB	387C	DB
3865	7E	386D	60	3875	FF	387D	C3
3866	3C	386E	30	3876	66	387E	66
3867	18	386F	00	3877	3C	387F	3C

16		17		18		19	
3880	FF	3888	3C	3890	3C	3898	3C
3881	C3	3889	7E	3891	66	3899	66
3882	C3	388A	DB	3892	C3	389A	C3
3883	FF	388B	DB	3893	DF	389B	FB
3884	C3	388C	DF	3894	DB	389C	DB
3885	C3	388D	C3	3895	DB	389D	DB
3886	C3	388E	66	3896	7E	389E	7E
3887	FF	388F	3C	3897	3C	389F	3C

20		21		22		23	
38A0	3C	38A8	00	38B0	7E	38B8	03
38A1	7E	38A9	01	38B1	66	38B9	03
38A2	DB	38AA	33	38B2	66	38BA	03
38A3	DB	38AB	1E	38B3	66	38BB	FF
38A4	FB	38AC	CE	38B4	66	38BC	03
38A5	C3	38AD	7B	38B5	66	38BD	03
38A6	66	38AE	31	38B6	66	38BE	03
38A7	3C	38AF	00	38B7	E7	38BF	00

24		25		26		27	
38C0	FF	38C8	18	38D0	3C	38D8	3C
38C1	66	38C9	18	38D1	66	38D9	66
38C2	3C	38CA	3C	38D2	66	38DA	C3
38C3	18	38CB	3C	38D3	30	38DB	FF
38C4	18	38CC	3C	38D4	18	38DC	C3
38C5	3C	38CD	3C	38D5	00	38DD	C3
38C6	66	38CE	18	38D6	18	38DE	66
38C7	FF	38CF	18	38D7	00	38DF	3C

```
     28                    29                    30                    31

38E0 ■■■■■■■■ FF      38E8 ■■■■■■■■ FF      38F0 ■■■■■■■■ FF      38F8 ■■■■■■■■ FF
38E1 ■■ ■■ ■■ DB      38E9 ■■    ■■ C3      38F1 ■■    ■■ C3      38F9 ■■ ■■ ■■ DB
38E2 ■■ ■■ ■■ DB      38EA ■■    ■■ C3      38F2 ■■    ■■ C3      38FA ■■ ■■ ■■ DB
38E3 ■■ ■■ ■■ DB      38EB ■■■■■ ■■ FB      38F3 ■■ ■■■■■ DF      38FB ■■ ■■ ■■ DB
38E4 ■■■■■ ■■ FB      38EC ■■ ■■ ■■ DB      38F4 ■■ ■■ ■■ DB      38FC ■■ ■■■■■ DF
38E5 ■■    ■■ C3      38ED ■■ ■■ ■■ DB      38F5 ■■ ■■ ■■ DB      38FD ■■    ■■ C3
38E6 ■■    ■■ C3      38EE ■■ ■■ ■■ DB      38F6 ■■ ■■ ■■ DB      38FE ■■    ■■ C3
38E7 ■■■■■■■■ FF      38EF ■■■■■■■■ FF      38F7 ■■■■■■■■ FF      38FF ■■■■■■■■ FF
```

```
     32                    33                    34                    35

3900          00      3908    ■■    18      3910  ■■ ■■   6C      3918  ■■ ■■   6C
3901          00      3909    ■■    18      3911  ■■ ■■   6C      3919  ■■ ■■   6C
3902          00      390A    ■■    18      3912  ■■ ■■   6C      391A ■■■■■■■  FE
3903          00      390B    ■■    18      3913          00      391B  ■■ ■■   6C
3904          00      390C    ■■    18      3914          00      391C ■■■■■■■  FE
3905          00      390D          00      3915          00      391D  ■■ ■■   6C
3906          00      390E    ■■    18      3916          00      391E  ■■ ■■   6C
3907          00      390F          00      3917          00      391F          00
```

```
     36                    37                    38                    39

3920    ■■    18      3928          00      3930   ■■■    38      3938    ■■    18
3921   ■■■■■  3E      3929 ■■   ■■  C6      3931  ■■ ■■   6C      3939    ■■    18
3922  ■ ■■    58      392A ■■  ■■   CC      3932   ■■■    38      393A   ■■     30
3923   ■■■■   3C      392B    ■■    18      3933  ■■■ ■■  76      393B          00
3924    ■■ ■  1A      392C   ■■     30      3934 ■■ ■■■   DC      393C          00
3925  ■■■■■  7C      392D  ■■  ■■  66      3935 ■■  ■■   CC      393D          00
3926    ■■    18      392E ■■   ■■  C6      3936  ■■■ ■■  76      393E          00
3927          00      392F          00      3937          00      393F          00
```

```
     40                    41                    42                    43

3940     ■■   0C      3948   ■■     30      3950          00      3958          00
3941    ■■    18      3949    ■■    18      3951  ■■  ■■  66      3959    ■■    18
3942   ■■     30      394A     ■■   0C      3952   ■■■■   3C      395A    ■■    18
3943   ■■     30      394B     ■■   0C      3953 ■■■■■■■■ FF      395B  ■■■■■■  7E
3944   ■■     30      394C     ■■   0C      3954   ■■■■   3C      395C    ■■    18
3945    ■■    18      394D    ■■    18      3955  ■■  ■■  66      395D    ■■    18
3946     ■■   0C      394E   ■■     30      3956          00      395E          00
3947          00      394F          00      3957          00      395F          00
```

```
     44                    45                    46                    47

3960          00      3968          00      3970          00      3978      ■■  06
3961          00      3969          00      3971          00      3979     ■■   0C
3962          00      396A          00      3972          00      397A    ■■    18
3963          00      396B  ■■■■■■  7E      3973          00      397B   ■■     30
3964          00      396C          00      3974          00      397C  ■■      60
3965    ■■    18      396D          00      3975    ■■    18      397D ■■       C0
3966    ■■    18      396E          00      3976    ■■    18      397E ■        80
3967   ■■     30      396F          00      3977          00      397F          00
```

```
     48                 49                 50                 51

3980  ■■■■■   7C  3988    ■■     18  3990   ■■■■   3C  3998   ■■■■   3C
3981 ■■   ■■  C6  3989   ■■■     38  3991  ■■  ■■  66  3999  ■■  ■■  66
3982 ■■  ■■■  CE  398A    ■■     18  3992      ■■  06  399A      ■■  06
3983 ■■ ■ ■■  D6  398B    ■■     18  3993   ■■■■   3C  399B    ■■■   1C
3984 ■■■  ■■  E6  398C    ■■     18  3994  ■■      60  399C      ■■  06
3985 ■■   ■■  C6  398D    ■■     18  3995  ■■  ■■  66  399D  ■■  ■■  66
3986  ■■■■■   7C  398E  ■■■■■■   7E  3996  ■■■■■■  7E  399E   ■■■■   3C
3987          00  398F           00  3997          00  399F          00
```

```
     52                 53                 54                 55

39A0    ■■■   1C  39A8  ■■■■■■   7E  39B0   ■■■■   3C  39B8  ■■■■■■  7E
39A1   ■■■■   3C  39A9  ■■   ■   62  39B1  ■■  ■■  66  39B9  ■■  ■■  66
39A2  ■■ ■■   6C  39AA  ■■       60  39B2  ■■      60  39BA      ■■  06
39A3 ■■  ■■   CC  39AB  ■■■■■    7C  39B3  ■■■■■   7C  39BB     ■■   0C
39A4 ■■■■■■■  FE  39AC      ■■   06  39B4  ■■  ■■  66  39BC    ■■    18
39A5     ■■   0C  39AD  ■■  ■■   66  39B5  ■■  ■■  66  39BD    ■■    18
39A6    ■■■■  1E  39AE   ■■■■    3C  39B6   ■■■■   3C  39BE    ■■    18
39A7          00  39AF           00  39B7          00  39BF          00
```

```
     56                 57                 58                 59

39C0   ■■■■   3C  39C8   ■■■■    3C  39D0          00  39D8          00
39C1  ■■  ■■  66  39C9  ■■  ■■   66  39D1          00  39D9          00
39C2  ■■  ■■  66  39CA  ■■  ■■   66  39D2    ■■    18  39DA    ■■    18
39C3   ■■■■   3C  39CB   ■■■■■   3E  39D3    ■■    18  39DB    ■■    18
39C4  ■■  ■■  66  39CC      ■■   06  39D4          00  39DC          00
39C5  ■■  ■■  66  39CD  ■■  ■■   66  39D5    ■■    18  39DD    ■■    18
39C6   ■■■■   3C  39CE   ■■■■    3C  39D6    ■■    18  39DE    ■■    18
39C7          00  39CF           00  39D7          00  39DF   ■■     30
```

```
     60                 61                 62                 63

39E0     ■■   0C  39E8           00  39F0  ■■      60  39F8   ■■■■   3C
39E1    ■■    18  39E9           00  39F1   ■■     30  39F9  ■■  ■■  66
39E2   ■■     30  39EA  ■■■■■■   7E  39F2    ■■    18  39FA  ■■  ■■  66
39E3  ■■      60  39EB           00  39F3     ■■   0C  39FB     ■■   0C
39E4   ■■     30  39EC           00  39F4    ■■    18  39FC    ■■    18
39E5    ■■    18  39ED  ■■■■■■   7E  39F5   ■■     30  39FD          00
39E6     ■■   0C  39EE           00  39F6  ■■      60  39FE    ■■    18
39E7          00  39EF           00  39F7          00  39FF          00
```

```
     64                 65                 66                 67

3A00  ■■■■■   7C  3A08    ■■     18  3A10 ■■■■■■   FC  3A18   ■■■■   3C
3A01 ■■   ■■  C6  3A09   ■■■■    3C  3A11  ■■  ■■  66  3A19  ■■  ■■  66
3A02 ■■ ■■■■  DE  3A0A  ■■  ■■   66  3A12  ■■  ■■  66  3A1A ■■       C0
3A03 ■■ ■■■■  DE  3A0B  ■■  ■■   66  3A13  ■■■■■   7C  3A1B ■■       C0
3A04 ■■ ■■■■  DE  3A0C  ■■■■■■   7E  3A14  ■■  ■■  66  3A1C ■■       C0
3A05 ■■       C0  3A0D  ■■  ■■   66  3A15  ■■  ■■  66  3A1D  ■■  ■■  66
3A06  ■■■■■   7C  3A0E  ■■  ■■   66  3A16 ■■■■■■   FC  3A1E   ■■■■   3C
3A07          00  3A0F           00  3A17          00  3A1F          00
```

```
     68                     69                     70                     71
3A20 ■■■■■    F8    3A28 ■■■■■■■  FE    3A30 ■■■■■■■  FE    3A38   ■■■■   3C
3A21  ■■ ■■   6C    3A29  ■■   ■  62    3A31  ■■   ■  62    3A39  ■■  ■■  66
3A22  ■■  ■■  66    3A2A  ■■ ■    68    3A32  ■■ ■    68    3A3A ■■       C0
3A23  ■■  ■■  66    3A2B  ■■■■    78    3A33  ■■■■    78    3A3B ■■       C0
3A24  ■■  ■■  66    3A2C  ■■ ■    68    3A34  ■■ ■    68    3A3C ■■  ■■■  CE
3A25  ■■ ■■   6C    3A2D  ■■   ■  62    3A35  ■■      60    3A3D  ■■  ■■  66
3A26 ■■■■■    F8    3A2E ■■■■■■■  FE    3A36 ■■■■     F0    3A3E   ■■■■■  3E
3A27          00    3A2F          00    3A37          00    3A3F          00

     72                     73                     74                     75
3A40  ■■  ■■  66    3A48  ■■■■■■  7E    3A50    ■■■■  1E    3A58 ■■■  ■■  E6
3A41  ■■  ■■  66    3A49    ■■    18    3A51     ■■   0C    3A59  ■■  ■■  66
3A42  ■■  ■■  66    3A4A    ■■    18    3A52     ■■   0C    3A5A  ■■ ■■   6C
3A43  ■■■■■■  7E    3A4B    ■■    18    3A53     ■■   0C    3A5B  ■■■■    78
3A44  ■■  ■■  66    3A4C    ■■    18    3A54 ■■  ■■   CC    3A5C  ■■ ■■   6C
3A45  ■■  ■■  66    3A4D    ■■    18    3A55 ■■  ■■   CC    3A5D  ■■  ■■  66
3A46  ■■  ■■  66    3A4E  ■■■■■■  7E    3A56  ■■■■    78    3A5E ■■■  ■■  E6
3A47          00    3A4F          00    3A57          00    3A5F          00

     76                     77                     78                     79
3A60 ■■■■     F0    3A68 ■■   ■■  C6    3A70 ■■   ■■  C6    3A78   ■■■    38
3A61  ■■      60    3A69 ■■■ ■■■  EE    3A71 ■■■  ■■  E6    3A79  ■■ ■■   6C
3A62  ■■      60    3A6A ■■■■■■■  FE    3A72 ■■■■ ■■  F6    3A7A ■■   ■■  C6
3A63  ■■      60    3A6B ■■■■■■■  FE    3A73 ■■ ■■■■  DE    3A7B ■■   ■■  C6
3A64  ■■   ■  62    3A6C ■■ ■ ■■  D6    3A74 ■■  ■■■  CE    3A7C ■■   ■■  C6
3A65  ■■  ■■  66    3A6D ■■   ■■  C6    3A75 ■■   ■■  C6    3A7D  ■■ ■■   6C
3A66 ■■■■■■■  FE    3A6E ■■   ■■  C6    3A76 ■■   ■■  C6    3A7E   ■■■    38
3A67          00    3A6F          00    3A77          00    3A7F          00

     80                     81                     82                     83
3A80 ■■■■■■   FC    3A88   ■■■    38    3A90 ■■■■■■   FC    3A98   ■■■■   3C
3A81  ■■  ■■  66    3A89  ■■ ■■   6C    3A91  ■■  ■■  66    3A99  ■■  ■■  66
3A82  ■■  ■■  66    3A8A ■■   ■■  C6    3A92  ■■  ■■  66    3A9A  ■■      60
3A83  ■■■■■   7C    3A8B ■■   ■■  C6    3A93  ■■■■■   7C    3A9B   ■■■■   3C
3A84  ■■      60    3A8C ■■ ■■ ■  DA    3A94  ■■ ■■   6C    3A9C      ■■  06
3A85  ■■      60    3A8D ■■  ■■   CC    3A95  ■■  ■■  66    3A9D  ■■  ■■  66
3A86 ■■■■     F0    3A8E  ■■■ ■■  76    3A96 ■■■  ■■  E6    3A9E   ■■■■   3C
3A87          00    3A8F          00    3A97          00    3A9F          00

     84                     85                     86                     87
3AA0  ■■■■■■  7E    3AA8  ■■  ■■  66    3AB0  ■■  ■■  66    3AB8 ■■   ■■  C6
3AA1  ■ ■■ ■  5A    3AA9  ■■  ■■  66    3AB1  ■■  ■■  66    3AB9 ■■   ■■  C6
3AA2    ■■    18    3AAA  ■■  ■■  66    3AB2  ■■  ■■  66    3ABA ■■   ■■  C6
3AA3    ■■    18    3AAB  ■■  ■■  66    3AB3  ■■  ■■  66    3ABB ■■ ■ ■■  D6
3AA4    ■■    18    3AAC  ■■  ■■  66    3AB4  ■■  ■■  66    3ABC ■■■■■■■  FE
3AA5    ■■    18    3AAD  ■■  ■■  66    3AB5   ■■■■   3C    3ABD ■■■ ■■■  EE
3AA6   ■■■■   3C    3AAE   ■■■■   3C    3AB6    ■■    18    3ABE ■■   ■■  C6
3AA7          00    3AAF          00    3AB7          00    3ABF          00
```

```
      88                        89                        90                        91

3AC0 ■■   ■■  C6   3AC8  ■■  ■■  66   3AD0 ■■■■■■■  FE   3AD8   ■■■■   3C
3AC1  ■■ ■■   6C   3AC9  ■■  ■■  66   3AD1 ■■   ■■  C6   3AD9   ■■     30
3AC2   ■■■    38   3ACA  ■■  ■■  66   3AD2 ■   ■■   8C   3ADA   ■■     30
3AC3   ■■■    38   3ACB   ■■■■   3C   3AD3    ■■    18   3ADB   ■■     30
3AC4  ■■ ■■   6C   3ACC    ■■    18   3AD4   ■■  ■  32   3ADC   ■■     30
3AC5 ■■   ■■  C6   3ACD    ■■    18   3AD5  ■■  ■■  66   3ADD   ■■     30
3AC6 ■■   ■■  C6   3ACE   ■■■■   3C   3AD6 ■■■■■■■  FE   3ADE   ■■■■   3C
3AC7          00   3ACF          00   3AD7          00   3ADF          00
```

```
      92                        93                        94                        95

3AE0 ■■       C0   3AE8   ■■■■   3C   3AF0    ■■    18   3AF8          00
3AE1  ■■      60   3AE9     ■■   0C   3AF1   ■■■■   3C   3AF9          00
3AE2   ■■     30   3AEA     ■■   0C   3AF2  ■■■■■■  7E   3AFA          00
3AE3    ■■    18   3AEB     ■■   0C   3AF3    ■■    18   3AFB          00
3AE4     ■■   0C   3AEC     ■■   0C   3AF4    ■■    18   3AFC          00
3AE5      ■■  06   3AED     ■■   0C   3AF5    ■■    18   3AFD          00
3AE6       ■  02   3AEE   ■■■■   3C   3AF6    ■■    18   3AFE          00
3AE7          00   3AEF          00   3AF7          00   3AFF ■■■■■■■■ FF
```

```
      96                        97                        98                        99

3B00   ■■     30   3B08          00   3B10 ■■■      E0   3B18          00
3B01    ■■    18   3B09          00   3B11  ■■      60   3B19          00
3B02     ■■   0C   3B0A  ■■■■    78   3B12  ■■■■■   7C   3B1A   ■■■■   3C
3B03          00   3B0B     ■■   0C   3B13  ■■  ■■  66   3B1B  ■■  ■■  66
3B04          00   3B0C  ■■■■■   7C   3B14  ■■  ■■  66   3B1C  ■■      60
3B05          00   3B0D ■■  ■■   CC   3B15  ■■  ■■  66   3B1D  ■■  ■■  66
3B06          00   3B0E  ■■■ ■■  76   3B16 ■■ ■■■   DC   3B1E   ■■■■   3C
3B07          00   3B0F          00   3B17          00   3B1F          00
```

```
      100                       101                       102                       103

3B20    ■■■   1C   3B28          00   3B30    ■■■   1C   3B38          00
3B21     ■■   0C   3B29          00   3B31   ■■ ■■  36   3B39          00
3B22  ■■■■■   7C   3B2A   ■■■■   3C   3B32   ■■     30   3B3A   ■■■■■  3E
3B23 ■■  ■■   CC   3B2B  ■■  ■■  66   3B33  ■■■■    78   3B3B  ■■  ■■  66
3B24 ■■  ■■   CC   3B2C  ■■■■■■  7E   3B34   ■■     30   3B3C  ■■  ■■  66
3B25 ■■  ■■   CC   3B2D  ■■      60   3B35   ■■     30   3B3D   ■■■■■  3E
3B26  ■■■ ■■  76   3B2E   ■■■■   3C   3B36  ■■■■    78   3B3E      ■■  06
3B27          00   3B2F          00   3B37          00   3B3F  ■■■■■   7C
```

```
      104                       105                       106                       107

3B40 ■■■      E0   3B48    ■■    18   3B50      ■■  06   3B58 ■■■      E0
3B41  ■■      60   3B49          00   3B51          00   3B59  ■■      60
3B42  ■■ ■■   6C   3B4A   ■■■    38   3B52     ■■■  0E   3B5A  ■■  ■■  66
3B43  ■■■ ■■  76   3B4B    ■■    18   3B53      ■■  06   3B5B  ■■ ■■   6C
3B44  ■■  ■■  66   3B4C    ■■    18   3B54      ■■  06   3B5C  ■■■■    78
3B45  ■■  ■■  66   3B4D    ■■    18   3B55  ■■  ■■  66   3B5D  ■■ ■■   6C
3B46 ■■■  ■■  E6   3B4E   ■■■■   3C   3B56  ■■  ■■  66   3B5E ■■■  ■■  E6
3B47          00   3B4F          00   3B57   ■■■■   3C   3B5F          00
```

108		109		110		111	
3B60	38	3B68	00	3B70	00	3B78	00
3B61	18	3B69	00	3B71	00	3B79	00
3B62	18	3B6A	6C	3B72	DC	3B7A	3C
3B63	18	3B6B	FE	3B73	66	3B7B	66
3B64	18	3B6C	D6	3B74	66	3B7C	66
3B65	18	3B6D	D6	3B75	66	3B7D	66
3B66	3C	3B6E	C6	3B76	66	3B7E	3C
3B67	00	3B6F	00	3B77	00	3B7F	00

112		113		114		115	
3B80	00	3B88	00	3B90	00	3B98	00
3B81	00	3B89	00	3B91	00	3B99	00
3B82	DC	3B8A	76	3B92	DC	3B9A	3C
3B83	66	3B8B	CC	3B93	76	3B9B	60
3B84	66	3B8C	CC	3B94	60	3B9C	3C
3B85	7C	3B8D	7C	3B95	60	3B9D	06
3B86	60	3B8E	0C	3B96	F0	3B9E	7C
3B87	F0	3B8F	1E	3B97	00	3B9F	00

116		117		118		119	
3BA0	30	3BA8	00	3BB0	00	3BB8	00
3BA1	30	3BA9	00	3BB1	00	3BB9	00
3BA2	7C	3BAA	66	3BB2	66	3BBA	C6
3BA3	30	3BAB	66	3BB3	66	3BBB	D6
3BA4	30	3BAC	66	3BB4	66	3BBC	D6
3BA5	36	3BAD	66	3BB5	3C	3BBD	FE
3BA6	1C	3BAE	3E	3BB6	18	3BBE	6C
3BA7	00	3BAF	00	3BB7	00	3BBF	00

120		121		122		123	
3BC0	00	3BC8	00	3BD0	00	3BD8	0E
3BC1	00	3BC9	00	3BD1	00	3BD9	18
3BC2	C6	3BCA	66	3BD2	7E	3BDA	18
3BC3	6C	3BCB	66	3BD3	4C	3BDB	70
3BC4	38	3BCC	66	3BD4	18	3BDC	18
3BC5	6C	3BCD	3E	3BD5	32	3BDD	18
3BC6	C6	3BCE	06	3BD6	7E	3BDE	0E
3BC7	00	3BCF	7C	3BD7	00	3BDF	00

124		125		126		127	
3BE0	18	3BE8	70	3BF0	76	3BF8	CC
3BE1	18	3BE9	18	3BF1	DC	3BF9	33
3BE2	18	3BEA	18	3BF2	00	3BFA	CC
3BE3	18	3BEB	0E	3BF3	00	3BFB	33
3BE4	18	3BEC	18	3BF4	00	3BFC	CC
3BE5	18	3BED	18	3BF5	00	3BFD	33
3BE6	18	3BEE	70	3BF6	00	3BFE	CC
3BE7	00	3BEF	00	3BF7	00	3BFF	33

```
     128                     129                     130                     131

3C00          00    3C08 ■■■■     F0    3C10     ■■■■ 0F    3C18 ■■■■■■■■ FF
3C01          00    3C09 ■■■■     F0    3C11     ■■■■ 0F    3C19 ■■■■■■■■ FF
3C02          00    3C0A ■■■■     F0    3C12     ■■■■ 0F    3C1A ■■■■■■■■ FF
3C03          00    3C0B ■■■■     F0    3C13     ■■■■ 0F    3C1B ■■■■■■■■ FF
3C04          00    3C0C          00    3C14          00    3C1C          00
3C05          00    3C0D          00    3C15          00    3C1D          00
3C06          00    3C0E          00    3C16          00    3C1E          00
3C07          00    3C0F          00    3C17          00    3C1F          00
```

```
     132                     133                     134                     135

3C20          00    3C28 ■■■■     F0    3C30     ■■■■ 0F    3C38 ■■■■■■■■ FF
3C21          00    3C29 ■■■■     F0    3C31     ■■■■ 0F    3C39 ■■■■■■■■ FF
3C22          00    3C2A ■■■■     F0    3C32     ■■■■ 0F    3C3A ■■■■■■■■ FF
3C23          00    3C2B ■■■■     F0    3C33     ■■■■ 0F    3C3B ■■■■■■■■ FF
3C24 ■■■■     F0    3C2C ■■■■     F0    3C34 ■■■■     F0    3C3C ■■■■     F0
3C25 ■■■■     F0    3C2D ■■■■     F0    3C35 ■■■■     F0    3C3D ■■■■     F0
3C26 ■■■■     F0    3C2E ■■■■     F0    3C36 ■■■■     F0    3C3E ■■■■     F0
3C27 ■■■■     F0    3C2F ■■■■     F0    3C37 ■■■■     F0    3C3F ■■■■     F0
```

```
     136                     137                     138                     139

3C40          00    3C48 ■■■■     F0    3C50     ■■■■ 0F    3C58 ■■■■■■■■ FF
3C41          00    3C49 ■■■■     F0    3C51     ■■■■ 0F    3C59 ■■■■■■■■ FF
3C42          00    3C4A ■■■■     F0    3C52     ■■■■ 0F    3C5A ■■■■■■■■ FF
3C43          00    3C4B ■■■■     F0    3C53     ■■■■ 0F    3C5B ■■■■■■■■ FF
3C44     ■■■■ 0F    3C4C     ■■■■ 0F    3C54     ■■■■ 0F    3C5C     ■■■■ 0F
3C45     ■■■■ 0F    3C4D     ■■■■ 0F    3C55     ■■■■ 0F    3C5D     ■■■■ 0F
3C46     ■■■■ 0F    3C4E     ■■■■ 0F    3C56     ■■■■ 0F    3C5E     ■■■■ 0F
3C47     ■■■■ 0F    3C4F     ■■■■ 0F    3C57     ■■■■ 0F    3C5F     ■■■■ 0F
```

```
     140                     141                     142                     143

3C60          00    3C68 ■■■■     F0    3C70     ■■■■ 0F    3C78 ■■■■■■■■ FF
3C61          00    3C69 ■■■■     F0    3C71     ■■■■ 0F    3C79 ■■■■■■■■ FF
3C62          00    3C6A ■■■■     F0    3C72     ■■■■ 0F    3C7A ■■■■■■■■ FF
3C63          00    3C6B ■■■■     F0    3C73     ■■■■ 0F    3C7B ■■■■■■■■ FF
3C64 ■■■■■■■■ FF    3C6C ■■■■■■■■ FF    3C74 ■■■■■■■■ FF    3C7C ■■■■■■■■ FF
3C65 ■■■■■■■■ FF    3C6D ■■■■■■■■ FF    3C75 ■■■■■■■■ FF    3C7D ■■■■■■■■ FF
3C66 ■■■■■■■■ FF    3C6E ■■■■■■■■ FF    3C76 ■■■■■■■■ FF    3C7E ■■■■■■■■ FF
3C67 ■■■■■■■■ FF    3C6F ■■■■■■■■ FF    3C77 ■■■■■■■■ FF    3C7F ■■■■■■■■ FF
```

```
     144                     145                     146                     147

3C80          00    3C88    ■■    18    3C90          00    3C98    ■■    18
3C81          00    3C89    ■■    18    3C91          00    3C99    ■■    18
3C82          00    3C8A    ■■    18    3C92          00    3C9A    ■■    18
3C83    ■■    18    3C8B    ■■    18    3C93    ■■■■■ 1F    3C9B    ■■■■■ 1F
3C84    ■■    18    3C8C    ■■    18    3C94    ■■■■■ 1F    3C9C     ■■■■ 0F
3C85          00    3C8D          00    3C95          00    3C9D          00
3C86          00    3C8E          00    3C96          00    3C9E          00
3C87          00    3C8F          00    3C97          00    3C9F          00
```

```
     148                  149                  150                  151

3CA0          00    3CA8    ■■    18    3CB0          00    3CB8    ■■    18
3CA1          00    3CA9    ■■    18    3CB1          00    3CB9    ■■    18
3CA2          00    3CAA    ■■    18    3CB2          00    3CBA    ■■    18
3CA3    ■■    18    3CAB    ■■    18    3CB3     ■■■■ 0F    3CBB    ■■■■■ 1F
3CA4    ■■    18    3CAC    ■■    18    3CB4    ■■■■■ 1F    3CBC    ■■■■■ 1F
3CA5    ■■    18    3CAD    ■■    18    3CB5    ■■    18    3CBD    ■■    18
3CA6    ■■    18    3CAE    ■■    18    3CB6    ■■    18    3CBE    ■■    18
3CA7    ■■    18    3CAF    ■■    18    3CB7    ■■    18    3CBF    ■■    18

     152                  153                  154                  155

3CC0          00    3CC8    ■■    18    3CD0          00    3CD8    ■■    18
3CC1          00    3CC9    ■■    18    3CD1          00    3CD9    ■■    18
3CC2          00    3CCA    ■■    18    3CD2          00    3CDA    ■■    18
3CC3 ■■■■■    F8    3CCB ■■■■■    F8    3CD3 ■■■■■■■■ FF    3CDB ■■■■■■■■ FF
3CC4 ■■■■■    F8    3CCC ■■■■     F0    3CD4 ■■■■■■■■ FF    3CDC ■■■■■■■■ FF
3CC5          00    3CCD          00    3CD5          00    3CDD          00
3CC6          00    3CCE          00    3CD6          00    3CDE          00
3CC7          00    3CCF          00    3CD7          00    3CDF          00

     156                  157                  158                  159

3CE0          00    3CE8    ■■    18    3CF0          00    3CF8    ■■    18
3CE1          00    3CE9    ■■    18    3CF1          00    3CF9    ■■    18
3CE2          00    3CEA    ■■    18    3CF2          00    3CFA    ■■    18
3CE3 ■■■■     F0    3CEB ■■■■■    F8    3CF3 ■■■■■■■■ FF    3CFB ■■■■■■■■ FF
3CE4 ■■■■■    F8    3CEC ■■■■■    F8    3CF4 ■■■■■■■■ FF    3CFC ■■■■■■■■ FF
3CE5    ■■    18    3CED    ■■    18    3CF5    ■■    18    3CFD    ■■    18
3CE6    ■■    18    3CEE    ■■    18    3CF6    ■■    18    3CFE    ■■    18
3CE7    ■■    18    3CEF    ■■    18    3CF7    ■■    18    3CFF    ■■    18

     160                  161                  162                  163

3D00    ■     10    3D08     ■■   0C    3D10  ■■  ■■  66    3D18   ■■■■   3C
3D01   ■■■    38    3D09    ■■    18    3D11  ■■  ■■  66    3D19  ■■  ■■  66
3D02  ■■ ■■   6C    3D0A   ■■     30    3D12          00    3D1A  ■■      60
3D03 ■■   ■■  C6    3D0B          00    3D13          00    3D1B ■■■■■    F8
3D04          00    3D0C          00    3D14          00    3D1C  ■■      60
3D05          00    3D0D          00    3D15          00    3D1D  ■■  ■■  66
3D06          00    3D0E          00    3D16          00    3D1E ■■■■■■■  FE
3D07          00    3D0F          00    3D17          00    3D1F          00

     164                  165                  166                  167

3D20   ■■■    38    3D28  ■■■■■■  7E    3D30    ■■■■  1E    3D38    ■■    18
3D21  ■   ■   44    3D29 ■■■■ ■   F4    3D31   ■■     30    3D39    ■■    18
3D22 ■ ■■■ ■  BA    3D2A ■■■■ ■   F4    3D32   ■■■    38    3D3A     ■■   0C
3D23 ■ ■   ■  A2    3D2B  ■■■ ■   74    3D33  ■■ ■■   6C    3D3B          00
3D24 ■ ■■■ ■  BA    3D2C   ■■ ■   34    3D34   ■■■    38    3D3C          00
3D25  ■   ■   44    3D2D   ■■ ■   34    3D35    ■■    18    3D3D          00
3D26   ■■■    38    3D2E   ■■ ■   34    3D36 ■■■■     F0    3D3E          00
3D27          00    3D2F          00    3D37          00    3D3F          00
```

168		169		170		171	
3D40	40	3D48	40	3D50	E0	3D58	00
3D41	C0	3D49	C0	3D51	10	3D59	18
3D42	44	3D4A	4C	3D52	62	3D5A	18
3D43	4C	3D4B	52	3D53	16	3D5B	7E
3D44	54	3D4C	44	3D54	EA	3D5C	18
3D45	1E	3D4D	08	3D55	0F	3D5D	18
3D46	04	3D4E	1E	3D56	02	3D5E	7E
3D47	00	3D4F	00	3D57	00	3D5F	00

172		173		174		175	
3D60	18	3D68	00	3D70	18	3D78	18
3D61	18	3D69	00	3D71	00	3D79	00
3D62	00	3D6A	00	3D72	18	3D7A	18
3D63	7E	3D6B	7E	3D73	30	3D7B	18
3D64	00	3D6C	06	3D74	66	3D7C	18
3D65	18	3D6D	06	3D75	66	3D7D	18
3D66	18	3D6E	00	3D76	3C	3D7E	18
3D67	00	3D6F	00	3D77	00	3D7F	00

176		177		178		179	
3D80	00	3D88	7C	3D90	00	3D98	3C
3D81	00	3D89	C6	3D91	66	3D99	60
3D82	73	3D8A	C6	3D92	66	3D9A	60
3D83	DE	3D8B	FC	3D93	3C	3D9B	3C
3D84	CC	3D8C	C6	3D94	66	3D9C	66
3D85	DE	3D8D	C6	3D95	66	3D9D	66
3D86	73	3D8E	F8	3D96	3C	3D9E	3C
3D87	00	3D8F	C0	3D97	00	3D9F	00

180		181		182		183	
3DA0	00	3DA8	38	3DB0	00	3DB8	00
3DA1	00	3DA9	6C	3DB1	C0	3DB9	00
3DA2	1E	3DAA	C6	3DB2	60	3DBA	66
3DA3	30	3DAB	FE	3DB3	30	3DBB	66
3DA4	7C	3DAC	C6	3DB4	38	3DBC	66
3DA5	30	3DAD	6C	3DB5	6C	3DBD	7C
3DA6	1E	3DAE	38	3DB6	C6	3DBE	60
3DA7	00	3DAF	00	3DB7	00	3DBF	60

184		185		186		187	
3DC0	00	3DC8	00	3DD0	03	3DD8	03
3DC1	00	3DC9	00	3DD1	06	3DD9	06
3DC2	00	3DCA	00	3DD2	0C	3DDA	0C
3DC3	FE	3DCB	7E	3DD3	3C	3DDB	66
3DC4	6C	3DCC	D8	3DD4	66	3DDC	66
3DC5	6C	3DCD	D8	3DD5	3C	3DDD	3C
3DC6	6C	3DCE	70	3DD6	60	3DDE	60
3DC7	00	3DCF	00	3DD7	C0	3DDF	C0

188		189		190		191	
3DE0	00	3DE8	00	3DF0	FE	3DF8	00
3DE1	E6	3DE9	00	3DF1	C6	3DF9	7C
3DE2	3C	3DEA	66	3DF2	60	3DFA	C6
3DE3	18	3DEB	C3	3DF3	30	3DFB	C6
3DE4	38	3DEC	DB	3DF4	60	3DFC	C6
3DE5	6C	3DED	DB	3DF5	C6	3DFD	6C
3DE6	C7	3DEE	7E	3DF6	FE	3DFE	EE
3DE7	00	3DEF	00	3DF7	00	3DFF	00

192		193		194		195	
3E00	18	3E08	18	3E10	00	3E18	00
3E01	30	3E09	0C	3E11	00	3E19	00
3E02	60	3E0A	06	3E12	00	3E1A	00
3E03	C0	3E0B	03	3E13	01	3E1B	80
3E04	80	3E0C	01	3E14	03	3E1C	C0
3E05	00	3E0D	00	3E15	06	3E1D	60
3E06	00	3E0E	00	3E16	0C	3E1E	30
3E07	00	3E0F	00	3E17	18	3E1F	18

196		197		198		199	
3E20	18	3E28	18	3E30	00	3E38	18
3E21	3C	3E29	0C	3E31	00	3E39	30
3E22	66	3E2A	06	3E32	00	3E3A	60
3E23	C3	3E2B	03	3E33	81	3E3B	C0
3E24	81	3E2C	03	3E34	C3	3E3C	C0
3E25	00	3E2D	06	3E35	66	3E3D	60
3E26	00	3E2E	0C	3E36	3C	3E3E	30
3E27	00	3E2F	18	3E37	18	3E3F	18

200		201		202		203	
3E40	18	3E48	18	3E50	18	3E58	C3
3E41	30	3E49	0C	3E51	3C	3E59	E7
3E42	60	3E4A	06	3E52	66	3E5A	7E
3E43	C1	3E4B	83	3E53	C3	3E5B	3C
3E44	83	3E4C	C1	3E54	C3	3E5C	3C
3E45	06	3E4D	60	3E55	66	3E5D	7E
3E46	0C	3E4E	30	3E56	3C	3E5E	E7
3E47	18	3E4F	18	3E57	18	3E5F	C3

204		205		206		207	
3E60	03	3E68	C0	3E70	CC	3E78	AA
3E61	07	3E69	E0	3E71	CC	3E79	55
3E62	0E	3E6A	70	3E72	33	3E7A	AA
3E63	1C	3E6B	38	3E73	33	3E7B	55
3E64	38	3E6C	1C	3E74	CC	3E7C	AA
3E65	70	3E6D	0E	3E75	CC	3E7D	55
3E66	E0	3E6E	07	3E76	33	3E7E	AA
3E67	C0	3E6F	03	3E77	33	3E7F	55

208		209		210		211	
3E80	FF	3E88	03	3E90	00	3E98	C0
3E81	FF	3E89	03	3E91	00	3E99	C0
3E82	00	3E8A	03	3E92	00	3E9A	C0
3E83	00	3E8B	03	3E93	00	3E9B	C0
3E84	00	3E8C	03	3E94	00	3E9C	C0
3E85	00	3E8D	03	3E95	00	3E9D	C0
3E86	00	3E8E	03	3E96	FF	3E9E	C0
3E87	00	3E8F	03	3E97	FF	3E9F	C0

212		213		214		215	
3EA0	FF	3EA8	FF	3EB0	01	3EB8	80
3EA1	FE	3EA9	7F	3EB1	03	3EB9	C0
3EA2	FC	3EAA	3F	3EB2	07	3EBA	E0
3EA3	F8	3EAB	1F	3EB3	0F	3EBB	F0
3EA4	F0	3EAC	0F	3EB4	1F	3EBC	F8
3EA5	E0	3EAD	07	3EB5	3F	3EBD	FC
3EA6	C0	3EAE	03	3EB6	7F	3EBE	FE
3EA7	80	3EAF	01	3EB7	FF	3EBF	FF

216		217		218		219	
3EC0	AA	3EC8	0A	3ED0	00	3ED8	A0
3EC1	55	3EC9	05	3ED1	00	3ED9	50
3EC2	AA	3ECA	0A	3ED2	00	3EDA	A0
3EC3	55	3ECB	05	3ED3	00	3EDB	50
3EC4	00	3ECC	0A	3ED4	AA	3EDC	A0
3EC5	00	3ECD	05	3ED5	55	3EDD	50
3EC6	00	3ECE	0A	3ED6	AA	3EDE	A0
3EC7	00	3ECF	05	3ED7	55	3EDF	50

220		221		222		223	
3EE0	AA	3EE8	AA	3EF0	01	3EF8	00
3EE1	54	3EE9	55	3EF1	02	3EF9	80
3EE2	A8	3EEA	2A	3EF2	05	3EFA	40
3EE3	50	3EEB	15	3EF3	0A	3EFB	A0
3EE4	A0	3EEC	0A	3EF4	15	3EFC	50
3EE5	40	3EED	05	3EF5	2A	3EFD	A8
3EE6	80	3EEE	02	3EF6	55	3EFE	54
3EE7	00	3EEF	01	3EF7	AA	3EFF	AA

224		225		226		227	
3F00	7E	3F08	7E	3F10	38	3F18	10
3F01	FF	3F09	FF	3F11	38	3F19	38
3F02	99	3F0A	99	3F12	FE	3F1A	7C
3F03	FF	3F0B	FF	3F13	FE	3F1B	FE
3F04	BD	3F0C	C3	3F14	FE	3F1C	7C
3F05	C3	3F0D	BD	3F15	10	3F1D	38
3F06	FF	3F0E	FF	3F16	38	3F1E	10
3F07	7E	3F0F	7E	3F17	00	3F1F	00

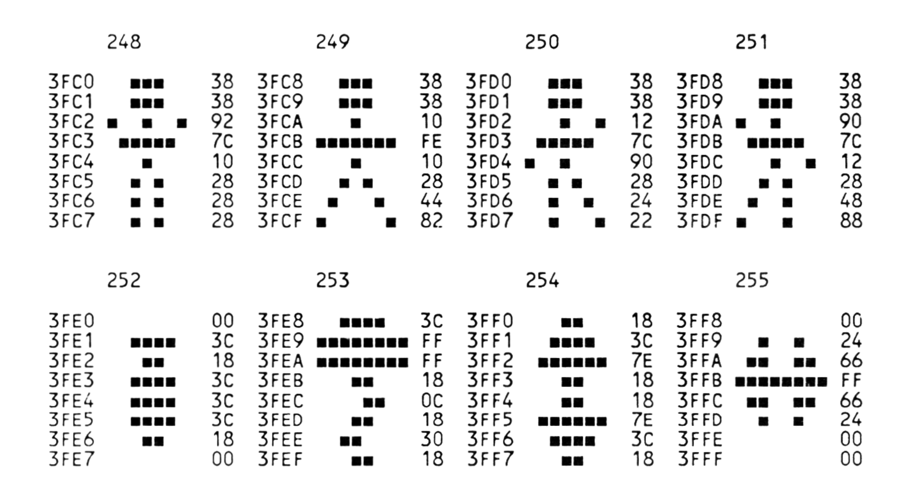
228		229		230		231	
3F20	6C	3F28	10	3F30	00	3F38	00
3F21	FE	3F29	38	3F31	3C	3F39	3C
3F22	FE	3F2A	7C	3F32	66	3F3A	7E
3F23	FE	3F2B	FE	3F33	C3	3F3B	FF
3F24	7C	3F2C	FE	3F34	C3	3F3C	FF
3F25	38	3F2D	10	3F35	66	3F3D	7E
3F26	10	3F2E	38	3F36	3C	3F3E	3C
3F27	00	3F2F	00	3F37	00	3F3F	00

232		233		234		235	
3F40	00	3F48	00	3F50	0F	3F58	3C
3F41	7E	3F49	7E	3F51	07	3F59	66
3F42	66	3F4A	7E	3F52	0D	3F5A	66
3F43	66	3F4B	7E	3F53	78	3F5B	66
3F44	66	3F4C	7E	3F54	CC	3F5C	3C
3F45	66	3F4D	7E	3F55	CC	3F5D	18
3F46	7E	3F4E	7E	3F56	CC	3F5E	7E
3F47	00	3F4F	00	3F57	78	3F5F	18

236		237		238		239	
3F60	0C	3F68	18	3F70	99	3F78	10
3F61	0C	3F69	1C	3F71	5A	3F79	38
3F62	0C	3F6A	1E	3F72	24	3F7A	38
3F63	0C	3F6B	1B	3F73	C3	3F7B	38
3F64	0C	3F6C	18	3F74	C3	3F7C	38
3F65	3C	3F6D	78	3F75	24	3F7D	38
3F66	7C	3F6E	F8	3F76	5A	3F7E	7C
3F67	38	3F6F	70	3F77	99	3F7F	D6

240		241		242		243	
3F80	18	3F88	18	3F90	10	3F98	08
3F81	3C	3F89	18	3F91	30	3F99	0C
3F82	7E	3F8A	18	3F92	70	3F9A	0E
3F83	FF	3F8B	18	3F93	FF	3F9B	FF
3F84	18	3F8C	FF	3F94	FF	3F9C	FF
3F85	18	3F8D	7E	3F95	70	3F9D	0E
3F86	18	3F8E	3C	3F96	30	3F9E	0C
3F87	18	3F8F	18	3F97	10	3F9F	08

244		245		246		247	
3FA0	00	3FA8	00	3FB0	80	3FB8	02
3FA1	00	3FA9	00	3FB1	E0	3FB9	0E
3FA2	18	3FAA	FF	3FB2	F8	3FBA	3E
3FA3	3C	3FAB	FF	3FB3	FE	3FBB	FE
3FA4	7E	3FAC	7E	3FB4	F8	3FBC	3E
3FA5	FF	3FAD	3C	3FB5	E0	3FBD	0E
3FA6	FF	3FAE	18	3FB6	80	3FBE	02
3FA7	00	3FAF	00	3FB7	00	3FBF	00



3.2 CARACTERES DE COMMANDE BASIC

Vous avez certainement déjà remarqué que certains caractères du jeu de caractères du CPC AMSTRAD ne peuvent être rendus visibles sur l'écran sans que cela pose quelques problèmes. Si vous faites par exemple tourner le programme suivant, vous constaterez que des choses assez curieuses se produisent :

```
10 FOR I=0 TO 255
20 PRINT CHR$(I);
30 NEXT I
```

Sur le plan de la logique pure, il semble que les 256 caractères disponibles auraient dû s'afficher proprement sur l'écran les uns à la suite des autres. Mais il n'en est rien : un bip retentit, l'écran se colore en rouge et quelque temps après le message "Ready" apparaît dans l'angle supérieur gauche de l'écran. La faute de ce résultat surprenant incombe aux caractères de codes 0 à 31, qu'on appelle les caractères de commande. Si on sort ces caractères avec PRINT CHR$, ils déclenchent des phénomènes prévus dans le système d'exploitation de l'ordinateur.

Isolons donc à titre d'exemple l'un de ces caractères de commande et voyons ce qui se passe.

PRINT CHR$(7)

Après que cette entrée ait été confirmée avec la touche RETURN, vous entendez à nouveau le son qui nous avait déjà surpris lors de notre première tentative. Après ce son, la file d'attente sonore est vidée. Ce caractère de commande fait partie de ceux qui sont tirés de la norme ASCII. Il peut être utilisé de façon très simple et profitable dans vos programmes personnels; il convient parfaitement pour signaler l'apparition d'une erreur.

De nombreux autres caractères de commande doivent d'abord être dotés d'autres paramètres pour pouvoir produire des résultats intéressants; certains sont rendus superflus par des instructions BASIC équivalentes. Pour plus de détails, étudiez la liste suivante qui vous présente tous les caractères de commande avec les applications possibles.

Chaque caractère de commande est suivi du nom qu'il porte dans le standard ASCII. Ce nom n'est pas toujours le plus adapté lorsqu'on utilise les caractères de commande sur le CPC mais nous l'avons malgré tout indiqué dans la liste pour pouvoir désigner les caractères de commande non seulement par leurs codes mais aussi par leurs noms.

CHR$(0) **NUL**

Aucun effet (Pointeur sur RET).

CHR$(1) **SOH**

Les caractères de 0 à 31 sont normalement toujours exécutés comme caractères de commande. Mais si on veut représenter sur l'écran un des caractères de cette zone au lieu de le faire exécuter comme caractère de commande, cela peut être obtenu avec CHR$(1).

Le programme suivant illustrera le fonctionnement du caractère de commande CHR$(1).

```
10 FOR CARACTERE=0 TO 31
20 PRINT CHR$(1);CHR$(CARACTERE);
30 NEXT CARACTERE
40 FOR CARACTERE=32 TO 255
50 PRINT CHR$(CARACTERE);
60 NEXT CARACTERE
```

Ce programme sort les 256 caractères du CPC sur l'écran. Dans la première boucle FOR-TO-NEXT, qui est chargée de la représentation des caractères 0 à 31, nous avons utilisé le code CHR$(1). Il entraîne la représentation graphique des caractères de commande.

La seconde boucle FOR-TO-NEXT ne sort que des caractères pouvant être représentés sans problème. Il n'est donc pas nécessaire de les faire précéder d'un CHR$(1).

CHR$(2) STX

L'exécution du CHR$(2) désactive le curseur de texte. L'effet de ce caractère de commande est identique à celui de l'instruction BASIC CURSOR avec un zéro comme paramètre pour le commutateur utilisateur.

Un INPUT a normalement pour effet de faire représenter le curseur dans l'emplacement de l'écran dans lequel l'entrée est attendue. Si vous ne voulez pas que le curseur soit représenté, vous pouvez utiliser CHR$(2) dans un tel cas. La ligne de programme suivante illustre ce principe :

```
10 PRINT CHR$(2):INPUT "VALEUR";VALEUR
```

La représentation du curseur a été interdite avec le code de commande CHR$(2).

CHR$(3) ETX

L'exécution de CHR$(3) active le curseur de texte. L'effet de ce caractère de commande est identique à celui de l'instruction BASIC CURSOR avec un 1 comme paramètre pour le commutateur utilisateur.

Si le curseur de texte a été désactivé à l'intérieur d'un programme BASIC avec CHR$(2), il peut être à nouveau activé avec CHR$(3).

```
10 PRINT CHR$(2):INPUT "PREMIERE VALEUR";PVALEUR
20 PRINT CHR$(3):INPUT "SECONDE VALEUR";SVALEUR
```

Le curseur de texte a été désactivé à la ligne 10 du programme, l'INPUT sera donc exécuté sans curseur (à cause de CHR$(2)). Pour le second INPUT en ligne 20, le curseur de texte apparaîtra à nouveau car il a été réactivé avec CHR$(3).

CHR$(4) EOT

L'effet de ce code de commande est le même que celui de l'instruction BASIC MODE X (X=0 à 2).

PRINT CHR$(4);CHR$(0)	correspond à	MODE 0
PRINT CHR$(4);CHR$(1)	correspond à	MODE 1
PRINT CHR$(4);CHR$(2)	correspond à	MODE 2

Remarquez que le paramètre utilisé pour CHR$(4) n'est pas la valeur entrée mais le reste de la division de cette valeur par quatre. De cette façon, il est possible d'entrer des paramètres supérieurs à 2 même si cela ne change rien au résultat obtenu.

```
PRINT CHR$(4);CHR$(1)
```

et

```
PRINT CHR$(4);CHR$(5)
```

auront donc la même signification et placeront tous deux le CPC en MODE 1.

CHR$(5) ENQ

Après un CHR$(5) le caractère suivant sera sorti dans la position actuelle du curseur graphique. Le programme suivant illustre son fonctionnement :

```
10 CLS
20 FOR I=0 TO 399
```

```
30 MOVE I,I
40 PRINT CHR$(5);".";
50 NEXT
```

Ce programme fait glisser sur l'écran un point du bas à gauche vers en haut à droite. Une boucle FOR-TO-NEXT produit à cet effet toutes les valeurs entières de 0 à 399. La ligne 30 déplace le curseur graphique avec ces valeurs comme paramètres. La ligne 40 place ensuite un point dans l'emplacement actuel du curseur graphique grâce à CHR$(5). Lors de chaque parcours de la boucle, la position du curseur graphique est modifiée et avec elle également l'emplacement de l'écran dans lequel apparaît le point.

CHR$(6) **ACK**

Ce caractère de commande active à nouveau l'écran de texte après qu'il ait été désactivé avec CHR$(21) (voir CHR$(21)).

```
10 PRINT CHR$(21)
20 FOR I=1 TO 40
30 PRINT "0";
40 NEXT
50 PRINT CHR$(6)
60 FOR I=1 TO 40
70 PRINT "1";
80 NEXT
```

La ligne 10 désactive d'abord l'écran de texte. Dans cet état, l'écran ne réagit plus à aucune entrée. Les lignes 20 à 40 sortiraient 40 zéros sur l'écran si l'écran était activé. Après un CHR$(21) cette représentation de caractères ne se fera pas.

La ligne 50 active enfin à nouveau l'écran de texte avec CHR$(6) de sorte que les caractères sortis par les lignes 60 à 80 apparaissent sur l'écran.

CHR$(7) **BEL**

PRINT CHR$(7) produit un son, le "bip" de l'ordinateur. Après exécution de ce son, la file d'attente des notes est vidée.

CHR$(8) **BS**

Le curseur est ramené d'un emplacement de caractère sur la gauche.

CHR$(9) **TAB**

Le curseur est avancé d'un emplacement de caractère sur la droite.

CHR$(10) **LF**

Le curseur descend d'une ligne.

CHR$(11) **VT**

Le curseur remonte d'une ligne.

CHR$(12) **FF**

PRINT CHR$(12) correspond à l'instruction BASIC CLS. Il vide la fenêtre de texte et place le curseur dans le coin supérieur gauche (HOME).

CHR$(13) **CR**

PRINT CHR$(13) correspond au fait d'appuyer sur la touche d'entrée (RETURN).

CHR$(14) **SO**

Le caractère de commande CHR$(14) a le même effet que l'instruction BASIC PAPER. La valeur suivante est utilisée comme paramètre pour la couleur, comme avec PAPER. La valeur fournie n'est toutefois pas utilisée directement comme paramètre mais seulement le reste de la division de cette valeur par 16.

CHR$(15) **SI**

Le caractère de commande CHR$(15) a le même effet que l'instruction BASIC PEN. La valeur suivante est utilisée comme paramètre pour la couleur, comme avec PEN. La valeur fournie n'est toutefois pas utilisée directement comme paramètre mais seulement le reste de la division de cette valeur par 16.

CHR$(16) **DLE**

Ce caractère de commande supprime le caractère dans l'emplacement du curseur et remplit le vide ainsi créé avec la couleur PAPER. Notez bien que CHR$(16) efface le caractère mais ne ramène pas les caractères pouvant éventuellement suivre ce caractère d'une position sur la gauche pour refermer le vide apparu.

CHR$(17) **DC1**

CHR$(17) efface sur la ligne actuelle tous les caractères du bord gauche de la fenêtre de texte jusqu'à l'emplacement du curseur (inclus). Le vide apparu est rempli avec la couleur PAPER comme pour CHR$(16).

CHR$(18) **DC2**

CHR$(18) efface sur la ligne actuelle tous les caractères à partir de l'emplacement du curseur (inclus) jusqu'au bord droit de la fenêtre de texte. Le vide apparu est rempli avec la couleur PAPER comme pour CHR$(16).

CHR$(19) **DC3**

CHR$(19) efface tous les caractères du coin supérieur gauche de la fenêtre de texte jusqu'à l'emplacement du curseur (inclus). Le vide apparu est rempli avec la couleur PAPER comme pour CHR$(16).

CHR$(20) DC4

CHR$(20) efface tous les caractères à partir de l'emplacement du curseur (inclus) jusqu'à la fin de la fenêtre de texte (coin inférieur droit). Le vide apparu est rempli avec la couleur PAPER comme pour CHR$(16).

CHR$(21) NAK

Ce caractère de commande désactive l'écran de texte (voir CHR$(6)). L'écran de texte ne réagit plus à aucune entrée après ce code de commande. CHR$(6) permet d'activer à nouveau l'écran de texte.

CHR$(22) SYN

CHR$(22) est un commutateur pour le mode transparent. Le programme suivant illustre bien ce que cela signifie :

```
10 CLS
20 PRINT CHR$(22);CHR$(1)
30 LOCATE 10,10
40 PRINT "AAAAAAAAAA"
50 LOCATE 10,10
60 PRINT "//////////"
70 PRINT CHR$(22);CHR$(0)
80 LOCATE 10,12
90 PRINT "AAAAAAAAAAA"
100 LOCATE 10,12
110 PRINT "///////////"
120 END
```

Si CHR$(22) est suivi du paramètre 1, le mode transparent est activé comme ici à la ligne 20. Les deux chaînes de caractères qui sont ensuite sorties dans le cours du programme dans le même emplacement de l'écran se superposent (mode transparent activé!).

CHR$(22) suivi d'une valeur de paramètre 0 désactive le mode transparent (ligne 70). Les deux mêmes chaînes de caractères que dans la première partie du programme ne se superposeront plus maintenant.

Les premiers caractères seront entièrement effacés par la seconde chaîne de caractères.

Ici également ce n'est pas la valeur entrée qui est utilisée directement comme paramètre mais le reste de la division de cette valeur par 2.

CHR$(23) ETB

CHR$(23) fixe le mode d'écriture graphique. Les valeurs de paramètre autorisées sont 0 à 3. Toute valeur supérieure est divisée par quatre et c'est son reste qui sera utilisé comme paramètre.

Signification du paramètre :

0 = Réglage normal
1 = Opération XOR entre premier plan et fond
2 = Opération AND entre premier plan et fond
3 = Opération OR entre premier plan et fond

Suivant le paramètre fixé, les sorties subiront après un CHR$(23) les opérations logiques XOR, AND ou OR avec le fond. L'opération logique entre le premier plan et le fond se fait bit par bit.

```
10 CLS
20 INPUT "PARAMETRE";P
30 TAG
40 MOVE 200,200
50 PRINT "**********";
60 TAGOFF
70 PRINT CHR$(23);CHR$(P)
80 TAG
90 MOVE 200,200
100 PRINT "##########";
110 TAGOFF
```

En ligne 20 la question "PARAMETRE?" vous demande d'entrer une valeur qui définira le type d'opération logique entre le premier plan et le fond (voir la table des paramètres). Les lignes 30 à 60 produisent un fond.

La ligne 70 fixe ensuite le type d'opération (d'après votre sélection). Une sortie est enfin effectuée sur le fond. Le premier plan et le fond sont combinés d'après les règles de l'opération logique sélectionnée.

CHR$(24) **CAN**

Le caractère de commande CHR$(24) échange les crayons de couleur de PAPER et PEN.

CHR$(25) **EM**

CHR$(25) correspond à l'instruction BASIC SYMBOL. Comme SYMBOL, CHR$(25) doit être complété par neuf paramètres dont les valeurs doivent être comprises entre 0 et 255.

La première valeur suivant le caractère de commande fixe la matrice de caractère qui devra être remplacée par une nouvelle matrice définie par l'utilisateur. Les huit paramètres suivants définissent alors la matrice.

```
10 SYMBOL AFTER 0
20 PRINT CHR$(25);
30 PRINT CHR$(123);
40 PRINT CHR$(&X11011011);        '■■ ■■ ■■
41 PRINT CHR$(&X00111100);        '  ■■■■
42 PRINT CHR$(&X01100110);        ' ■■  ■■
43 PRINT CHR$(&X01100110);        ' ■■  ■■
44 PRINT CHR$(&X01111110);        ' ■■■■■■
45 PRINT CHR$(&X01100110);        ' ■■  ■■
46 PRINT CHR$(&X01100110);        ' ■■  ■■
47 PRINT CHR$(&X00000000);        '
```

Les commentaires placés en fin des lignes 40 à 47 servent uniquement à permettre au lecteur de mieux comprendre le principe utilisé. Vous ne devez pas les taper.

Ce programme redéfinit le caractère 123 ({) en un "Ä".

Nous avons choisi la notation binaire pour les huit derniers paramètres pour que vous saisissiez plus aisément le rapport entre ces paramètres et la matrice de caractère. Chaque bit mis dans ces huit paramètres correspond à un point allumé dans la matrice du caractère.

CHR$(26) **SUB**

Le caractère de commande CHR$(26) correspond à l'instruction BASIC WINDOW (sans indication d'un canal). Ce code de commande est suivi de quatre paramètres qui fixent la taille de la fenêtre. Les deux premiers paramètres fixent les bords gauche et droit de la fenêtre. L'ordre de l'entrée est sans importance, c'est automatiquement la plus petite des deux valeurs qui sera prise comme limite gauche de la fenêtre, la plus grande valeur étant prise comme limite droite.

Il en va de même pour les deux derniers paramètres. Le plus petit des deux fixe la limite supérieure, le plus grand la limite inférieure de la fenêtre.

CHR$(27) **ESC**

Aucun effet (Pointeur sur RET).

CHR$(28) **FS**

Le code de commande CHR$(28) correspond à l'instruction BASIC INK. Comme INK, ce caractère de commande doit être suivi de trois paramètres. Le premier paramètre ou le reste d'une division de ce paramètre par 16 fixe le crayon de couleur qui sera modifié par les indications de couleur suivantes. Le reste des deux paramètres suivants divisés par 32 fixe les deux couleurs pour le crayon de couleur voulu.

Exemple :

```
PRINT CHR$(28);CHR$(1);CHR$(0);CHR$(13)
```

correspond à

```
INK 1,0,13
```

CHR$(29) **GS**

Le caractère de commande CHR$(29) correspond à l'instruction BASIC BORDER. Le caractère de commande est suivi de deux paramètres qui représentent les deux valeurs de couleur pour le bord de l'écran.

Pour déterminer les valeurs de couleur, les paramètres ne sont pas utilisés directement mais c'est leur reste après division par 32 qui est utilisé.

CHR$(30) **RS**

Le code de commande CHR$(30) place le curseur dans le coin supérieur gauche de la fenêtre de texte (HOME).

CHR$(31) **US**

Comme l'instruction BASIC LOCATE, le caractère de commande CHR$(31) place le curseur dans l'emplacement indiqué de la fenêtre de texte. L'emplacement sur lequel le curseur doit être amené est fixé par les deux paramètres suivant le caractère de commande. Le premier paramètre indique la position de colonne, le second la position de ligne. Si les paramètres sont supérieurs aux limites de la fenêtre, le curseur est placé dans l'emplacement autorisé le plus près des valeurs indiquées.

3.3 PLUSIEURS CARACTERES FORMENT UNE IMAGE

Maintenant que vous avez étudié le jeu de caractères complet du CPC et que vous savez comment venir à bout des caractères de commande les plus coriaces, nous allons vous présenter une méthode simple pour réaliser des dessins à partir des caractères disponibles.

Cette méthode est aussi simple que possible : on prend une image qui servira de modèle, cela peut être un modèle que vous aurez réalisé vous-même ou bien une image toute faite, et on commence par se poser quelques questions de base. Il faut d'abord fixer la taille que l'image devra avoir plus tard sur l'écran et en fonction de cette taille il faut décomposer le modèle en éléments d'image

carrés correspondant à la taille d'un caractère. Une fois que le modèle aura été préparé de cette façon, il faut trouver pour chaque élément de l'image un caractère qui lui corresponde le plus possible. Il ne reste plus alors qu'à faire envoyer les différents caractères sur l'écran par un programme approprié et votre dessin est terminé.

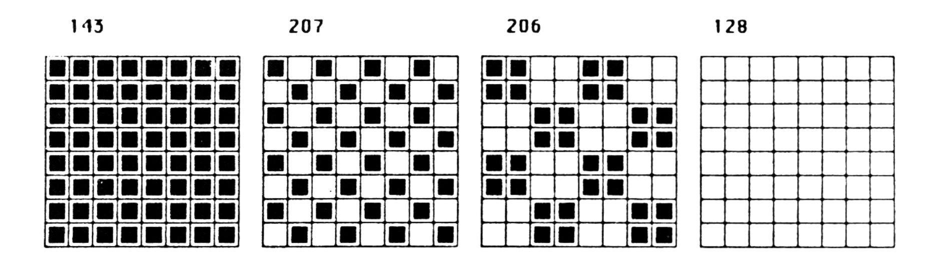
Voyez maintenant comment appliquer ces explications théoriques dans la pratique! Nous allons prendre pour modèle pour le dessin à réaliser la figure 4.



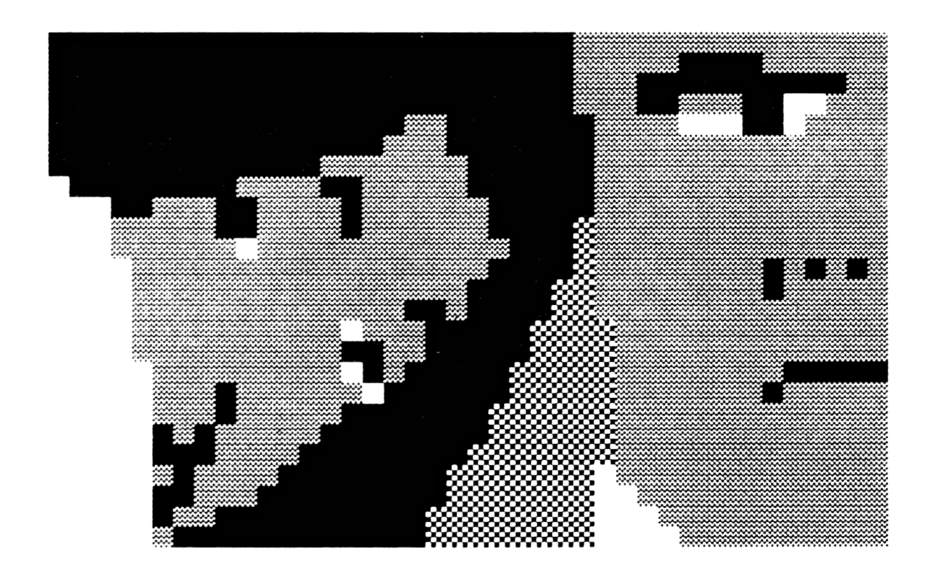
Figure 4 : **Modèle pour le dessin à base de caractères**

Comme cette illustration ne représente pas un détail tel qu'on pourrait en rencontrer dans la programmation de jeux et qu'il s'agit d'un motif complet, il faudra qu'il remplisse l'écran tout entier. Si l'on travaille en MODE 1, cela signifie que le modèle devra être divisé en 40 fois 25 éléments d'image. Pour chacun des mille éléments d'image il faudra maintenant trouver un caractère qui le restitue le mieux possible. Pour que ce travail ne dégénère pas en un travail de titan, nous nous sommes limités dans la digitalisation du dessin à diverses nuances de gris, ce qui ne contredit en rien le principe utilisé.

Les différentes nuances de gris sont produites avec les caractères 143 (noir), 207 (gris), 206 (gris clair) et 128 (blanc).



Après qu'ait été trouvé un caractère correspondant à chaque élément d'image du modèle, les codes de caractère de ces caractères doivent être rassemblés dans des lignes DATA. En réalisant les instructions DATA, nous avons veillé à ce que les codes de caractère correspondant à des caractères qui figureront plus tard sur une même ligne de l'écran soient toujours rassemblés sous le même numéro de ligne. Chacune des 25 lignes DATA comporte donc précisément 40 codes de caractère ce qui facilite considérablement la détection de l'origine d'erreurs de frappe éventuelles.



Copie d'écran 2 : **Plusieurs caractères forment une image**

La méthode qui consiste à faire apparaître le dessin sur l'écran à partir des mille DATAs dans lesquelles est codifié le modèle de l'image est un exercice tel qu'on peut en faire dans la deuxième séance d'un cours de BASIC. Les DATAs sont lues et envoyées sur l'écran avec PRINT CHR$ dans une boucle FOR-TO-NEXT qui est parcourue 1000 fois très précisément. L'image reproduite par la copie d'écran 2 est ainsi construite du haut à gauche au bas à droite.

```
100 MODE 1
110 INK 1,0
120 INK 0,13
130 '
140 FOR i=1 TO 1000
150 READ d
160 PRINT CHR$(D);
170 NEXT i
180 '
```

```
190 DATA 143,143,143,143,143,143,143,143,143,143,143,143,14
3,143,143,143,143,143,143,143,143,143,143,143,143,207,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
200 DATA 143,143,143,143,143,143,143,143,143,143,143,143,14
3,143,143,143,143,143,143,143,143,143,143,143,143,207,207,2
07,207,207,143,143,143,143,207,207,207,207,207,207
210 DATA 143,143,143,143,143,143,143,143,143,143,143,143,14
3,143,143,143,143,143,143,143,143,143,143,143,143,207,207,2
07,143,143,143,143,143,143,143,143,143,143,207,207
220 DATA 143,143,143,143,143,143,143,143,143,143,143,143,14
3,143,143,143,143,143,143,143,143,143,143,143,143,207,207,2
07,143,143,207,207,207,143,143,128,128,207,207,207
230 DATA 143,143,143,143,143,143,143,143,143,143,143,143,14
3,143,143,143,143,207,207,143,143,143,143,143,143,143,207,2
07,207,207,128,128,128,143,143,128,207,207,207,207
240 DATA 143,143,143,143,143,143,143,143,143,143,143,143,14
3,143,143,143,207,207,207,143,143,143,143,143,143,143,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
250 DATA 143,143,143,143,143,143,143,143,143,143,143,143,14
3,207,207,207,207,207,207,207,143,143,143,143,143,143,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
260 DATA 128,143,143,143,143,143,143,143,143,207,207,207,20
7,143,143,207,207,207,207,207,143,143,143,143,143,143,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
270 DATA 128,128,128,143,143,207,207,207,143,143,207,207,20
7,207,143,207,207,207,207,207,207,143,143,143,143,143,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
280 DATA 128,128,128,207,207,207,207,207,143,143,207,207,20
7,207,143,207,207,207,207,207,207,143,143,143,143,206,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
290 DATA 128,128,128,207,207,207,207,207,207,128,207,207,20
7,207,207,207,207,207,207,207,207,207,143,143,143,206,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
300 DATA 128,128,128,128,207,207,207,207,207,207,207,207,20
7,207,207,207,207,207,207,207,207,143,143,143,143,206,207,2
07,207,207,207,207,207,207,143,207,143,207,143,207
310 DATA 128,128,128,128,207,207,207,207,207,207,207,207,20
7,207,207,207,207,207,207,207,143,143,143,143,206,206,207,2
07,207,207,207,207,207,207,143,207,207,207,207,207
320 DATA 128,128,128,128,207,207,207,207,207,207,207,207,20
7,207,207,207,207,143,143,207,143,143,143,143,206,206,207,2
07,207,207,207,207,207,207,207,207,207,207,207,207
```

```
330 DATA 128,128,128,128,207,207,207,207,207,207,207,207,20
7,207,128,207,207,207,143,143,143,143,143,206,206,206,206,2
07,207,207,207,207,207,207,207,207,207,207,207,207
340 DATA 128,128,128,128,207,207,207,207,207,207,207,207,20
7,207,143,143,207,207,143,143,143,143,143,206,206,206,206,2
07,207,207,207,207,207,207,207,207,207,207,207,207
350 DATA 128,128,128,128,128,207,207,207,207,207,207,207,20
7,207,128,143,207,143,143,143,143,143,206,206,206,206,206,2
07,207,207,207,207,207,207,207,143,143,143,143,143
360 DATA 128,128,128,128,128,207,207,207,143,207,207,207,20
7,207,207,128,143,143,143,143,143,143,206,206,206,206,206,2
07,207,207,207,207,207,207,143,207,207,207,207,207
370 DATA 128,128,128,128,128,207,207,207,143,207,207,207,20
7,207,143,143,143,143,143,143,143,206,206,206,206,206,206,2
07,207,207,207,207,207,207,207,207,207,207,207,207
380 DATA 128,128,128,128,128,143,207,143,207,207,207,207,20
7,143,143,143,143,143,143,143,143,206,206,206,206,206,206,2
07,207,207,207,207,207,207,207,207,207,207,207,207
390 DATA 128,128,128,128,128,143,143,143,207,207,207,207,14
3,143,143,143,143,143,143,143,206,206,206,206,206,206,206,2
07,207,207,207,207,207,207,207,207,207,207,207,207
400 DATA 128,128,128,128,128,207,143,207,207,207,207,143,14
3,143,143,143,143,143,143,206,206,206,206,206,206,206,128,2
07,207,207,207,207,207,207,207,207,207,207,207,207
410 DATA 128,128,128,128,128,143,143,207,207,207,143,143,14
3,143,143,143,143,143,143,206,206,206,206,206,206,206,128,1
28,207,207,207,207,207,207,207,207,207,207,207,207
420 DATA 128,128,128,128,128,143,207,207,143,143,143,143,14
3,143,143,143,143,143,206,206,206,206,206,206,206,206,128,1
28,128,207,207,207,207,207,207,207,207,207,207,207
430 DATA 128,128,128,128,128,207,143,143,143,143,143,143,14
3,143,143,143,143,143,206,206,206,206,206,206,206,206,128,1
28,128,128,207,207,207,207,207,207,207,207,207,207
440 '
450 GOTO 450
```

3.4 L'EDITEUR DE JEU DE CARACTERES

Nous avons jusqu'ici toujours considéré le jeu de caractères comme quelque chose d'immuable dont les éléments sont fixés à l'intérieur de l'ordinateur sans qu'il soit possible d'y toucher. Nous allons cependant vous montrer maintenant qu'il est tout à fait possible de transformer entièrement les différents caractères et nous vous fournirons en même temps un outil vous permettant de le faire.

Nous vous avons déjà indiqué qu'un caractère déterminé se compose de huit fois huit points d'image, chaque point contribuant à définir l'apparence d'un caractère. Nous allons maintenant vous montrer comment un caractère peut être redéfini pour recevoir n'importe quelle autre forme. Nous vous demandons pour cela de bien vouloir prendre une feuille de papier quadrillé et un crayon.

Délimitez sur le papier, avec le crayon, une surface carrée de huit fois huit cases. Dessinez alors un caractère quelconque dans cette surface. Peu importe qu'il s'agisse d'un signe reconnaissable ou d'un caractère totalement imaginaire. Coloriez alors les cases les plus importantes pour l'apparence du caractère. Vous aurez ainsi déjà une première idée de l'apparence qu'aura plus tard ce caractère sur l'écran.

La prochaine étape consiste à traduire ce caractère sous une forme compréhensible pour l'ordinateur. Il vous faut pour cela examiner isolément chaque ligne de la matrice que vous avez dessinée sur le papier et écrire à côté de chaque ligne une série de zéros et de uns. Vous écrirez un zéro pour une case non coloriée et un un pour une case coloriée. Une fois ce travail réalisé pour les huit lignes de la matrice de caractère, votre feuille devra se présenter, dans le principe, de la même façon que la figure 5, page suivante.

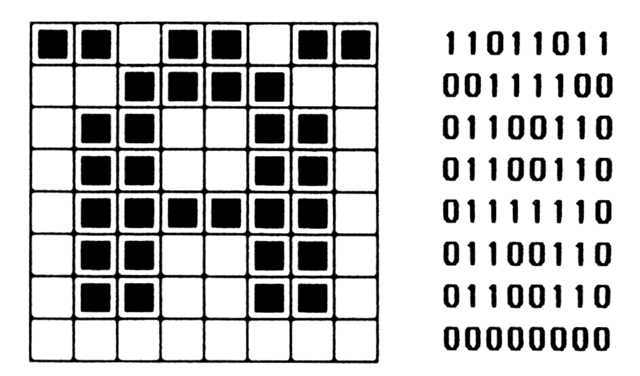


Figure 5 : **Définition de caractère à la main**

Maintenant que le caractère existe tout au moins sur le papier, reste la question de savoir comment placer ce caractère dans l'ordinateur. Les deux instructions du BASIC Locomotive

```
SYMBOL AFTER x
```

et

```
SYMBOL x,r1,r2,r3,r4,r5,r6,r7,r8
```

sont prévues spécialement pour cela. Pour comprendre le mode de fonctionnement de ces instructions, il faut choisir un caractère quelconque qui devra être remplacé par la nouvelle matrice de caractère. Dans notre exemple, nous remplacerons le caractère numéro 123 ({) par notre création. La marche à suivre est aussi simple que possible. SYMBOL AFTER 123 vous permet de redéfinir les caractères à partir du numéro de code 123. SYMBOL 123,r1,r2,r3,r4, r5,r6,r7,r8 vous permet ensuite d'intégrer la matrice réalisée dans le jeu de caractères. Vous devez écrire à la place de r1 à r8 les lignes de zéros et de uns que vous avez écrites sur le papier à côté de la matrice de caractère. Notez bien qu'il s'agit d'un mode d'écriture binaire de nombres et que chaque valeur doit par conséquent être précédée d'un "&X". Appliqué à l'exemple que nous vous donnons dans le livre, cela donnerait le programme suivant :

```
10 SYMBOL AFTER 123
20 SYMBOL 123,&X11011011,&X00111100,&X01100110,&X01100110,&
X01111110,&X01100110,&X01100110,&X00000000
```

Une fois que ce programme a été exécuté, vous pouvez essayer le caractère que vous venez de définir. Il vous suffit pour cela de l'envoyer sur l'écran avec PRINT CHR$(code de caractère)!

La méthode que nous avons choisie pour redéfinir un caractère ne constitue qu'un exemple parmi bien d'autres voies possibles mais elle montre bien en tout cas tout le travail que nécessitent la définition d'un seul caractère et son intégration par l'ordinateur. Vous pouvez facilement imaginer le travail que représenterait dans ces conditions la définition d'un jeu de caractères entier.

La fonction principale d'un ordinateur est cependant d'accélérer les processus de travail. Il serait donc bienvenu de pouvoir se décharger sur le CPC d'une partie du long travail de définition de caractères. Pour que le CPC puisse remplir cette tâche, il faudrait écrire un programme permettant d'éditer (c'est-à-dire d'examiner et de modifier) le jeu de caractères. Quelles possibilités devrait offrir un tel éditeur du jeu de caractères ?

1. L'éditeur devra disposer d'une zone d'édition comparable à la surface de huit fois huit cases que vous avez dessinée sur le papier. Il faut qu'on puisse charger et modifier une matrice de caractère dans cette zone.

2. Le programme doit posséder un curseur d'édition qui puisse être déplacé à l'intérieur des limites de la zone d'édition à l'aide des touches curseur.

3. Pour modifier le caractère dans la zone d'édition, il faut qu'on puisse fixer ou effacer un point image dans l'emplacement actuel du curseur. Cette opération équivaut au fait de colorier ou de gommer une case sur le papier.

4. L'éditeur doit permettre d'intégrer les caractères achevés dans le jeu de caractères actuel du CPC pour qu'ils soient dorénavant à la disposition de l'utilisateur.

5. Pour que les caractères une fois modifiés puissent être conservés pour pouvoir aussi être utilisés ultérieurement, il faut qu'on puisse écrire les caractères modifiés sur disquette. Il faut pour cela que les caractères soient stockés sous forme d'un fichier de programme prêt à fonctionner qui pourra en cas de besoin être chargé à la suite d'autres parties de programme mais qui pourra aussi fonctionner de façon indépendante.

En partant de ces exigences, nous avons écrit pour vous un éditeur de jeu de caractères dont nous allons vous expliquer d'abord le mode d'emploi avant de vous présenter le listing du programme.

Après que le programme ait été lancé avec RUN, une surface de huit fois huit cases est encadrée sur l'écran, surface qui fonctionnera plus tard comme zone d'édition. A droite de cette zone sont sorties les options du programme avec les méthodes permettant de les appeler. Dans l'angle supérieur gauche (position par défaut) de la zone d'édition apparaît le curseur d'édition. Il a la forme d'un caractère plein marbré et il peut être déplacé librement avec les touches curseur à l'intérieur de la zone d'édition. Si lors de son déplacement le curseur bute contre le cadre de la zone, un son retentit pour indiquer que la position voulue n'est pas autorisée.

Des indications sonores d'erreur retentissent chaque fois que vous demandez au programme quelque chose qui n'a pas de sens ou qui est impossible. Si vous appuyez par exemple, à ce moment précis, sur n'importe quelle touche autre que "N" et "Q", cela entraînera un message d'erreur (un bip sonore). En appuyant sur la touche "Q", vous mettrez fin au programme. Si vous actionnez par contre la touche "N" pour NOUVEAU CARACTERE, le programme vous demandera d'entrer un code de caractère. Une fois qu'un numéro de code aura été entré, confirmé avec RETURN et accepté par le programme, c'est-à-dire qu'il n'y aura pas eu de message d'erreur provoqué par un nombre sortant des limites autorisées, alors l'éditeur sortira dans le coin supérieur gauche de l'écran le caractère correspondant à votre entrée. Il sera alors recopié agrandi dans la zone d'édition à partir de cet emplacement. Il sera alors à nouveau effacé de la position HOME (coin supérieur gauche de l'écran). Une fois que vous serez parvenu jusqu'à ce point dans l'utilisation de l'éditeur, vous pourrez appeler toutes les options affichées. Examinons les autres options dans le détail :

F : En appuyant sur la touche "F", vous fixez le point image à l'emplacement actuel du curseur d'édition.

S : En actionnant la touche "S" vous supprimez au contraire le point image se trouvant à la position du curseur.

I : Cette option vous permet de faire intégrer dans le jeu de caractères un caractère tel qu'il figure dans la zone d'édition. Une fois que vous avez modifié un caractère, vous ne devez bien sûr pas vous étonner de le voir apparaître au beau milieu d'un message affiché par l'éditeur.

Q : L'option QUITTER LE PROGRAMME n'est pas aussi simple qu'on pourrait le penser à première vue car elle ne met fin au programme que si

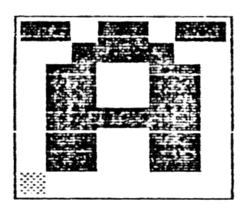
1. soit aucun caractère n'a été sélectionné pour être édité (Option N).

2. soit aucun des caractères édités n'a été intégré dans le jeu de caractères (Option I).

Si un caractère a été intégré dans le jeu de caractères, lorsque vous appuyez sur la touche "Q", la question suivante apparaîtra : **"Voulez-vous sauver les caractères édités sur disquette sous forme d'un fichier programme (O/N)"**. Si vous répondez par non à cette question, le programme est interrompu et les caractères redéfinis ne seront disponibles que dans la mémoire où ils seront conservés jusqu'à ce que la mémoire soit vidée ou modifiée. Si vous répondez par oui à la question, le programme vous demandera d'entrer un nom de fichier après quoi il écrira les caractères édités sur disquette. Le programme rencontre ensuite END et s'interrompt.

Vous pouvez, si vous le voulez, reprendre comme premier essai d'utilisation du programme l'exemple de définition d'un a tréma majuscule avec lequel nous avons illustré la définition manuelle d'un caractère. Après lancement du programme, actionnez la touche "N" et entrez comme code de caractère le nombre 123, comme dans

notre exemple. Le programme reproduira alors dans la zone d'édition l'image d'un crochet "{". C'est en effet ce caractère qui correspond normalement à ce code. Vous pouvez maintenant réaliser avec les touches curseur et les options FIXER POINT et SUPPRIMER POINT le a tréma majuscule que vous commencez certainement à bien connaître. Une fois la définition terminée, l'écran devra se présenter comme sur la copie d'écran 3.



↑↓←→: DEPLACER CURSEUR
F: FIXER POINT
S: SUPPRIMER POINT
N: NOUVEAU CARACTERE
I: INTEGRER
Q: QUITTER PROGRAMME

Copie d'écran 3 : **Définition d'un a tréma majuscule**

Vous appuyez ensuite sur la touche "I" pour faire intégrer dans le jeu de caractères du CPC le a tréma majuscule à la place du crochet "{". Si vous chargez ensuite le fichier de programme qui aura été écrit sur disquette par l'option QUITTER LE PROGRAMME et si vous le faites afficher sur l'écran avec LIST, il devra se présenter ainsi :

```
1 'FICHIER PROGRAMME CREE AUTOMATIQUEMENT
2 SYMBOL AFTER 0
10 SYMBOL 123, 219, 60, 102, 102, 126, 102, 102, 0
```

Vous pouvez charger et faire tourner ce programme n'importe quand, il intégrera toujours un a tréma majuscule dans votre ordinateur. Vous voyez qu'il est beaucoup plus facile de concevoir et de modifier un caractère avec l'éditeur de jeu de caractères. Le travail avec l'éditeur de jeu de caractères se révèle particulièrement pratique lorsqu'on veut modifier de nombreux caractères comme on est amené à le faire pour beaucoup d'applications dans le domaine de la programmation graphique.

```
100 '************************************************************
110 '***                                                      ***
120 '***    EDITEUR DE JEU DE CARACTERES  JST 21.8.1986       ***
130 '***                                                      ***
140 '************************************************************
150 '
160 MODE 1
170 SYMBOL AFTER 0
180 WINDOW#1,1,40,22,25
190 '
200 DIM CAR(255,7)
210 DIM EZ(255)
220 '
230 CX=5
240 CY=9
250 '
260 'DESSINER LE CADRE DE LA ZONE D'EDITION
270 '
280 MOVE 60,141
290 DRAWR 134,0
300 DRAWR 0,134
310 DRAWR -134,0
320 DRAWR 0,-134
330 '
340 'SORTIR LE MENU
350 '
360 LOCATE 16,9:PRINT CHR$(240);CHR$(241);CHR$(242);CHR$(243);": DE
PLACER CURSEUR"
370 LOCATE 16,11:PRINT "     F: FIXER POINT"
380 LOCATE 16,12:PRINT "     S: SUPPRIMER POINT"
390 LOCATE 16,14:PRINT "     N: NOUVEAU CARACTERE"
400 LOCATE 16,15:PRINT "     I: INTEGRER"
410 LOCATE 16,16:PRINT "     Q: QUITTER PROGRAMME"
420 '
430 GOSUB 650
440 '
450 'BOUCLE PRINCIPALE
460 '
470 B=1
480 '
490 E$=UPPER$(INKEY$)
500 IF E$="" THEN 490
```

```
510 IF E$=CHR$(240) OR E$=CHR$(241) OR E$=CHR$(242) OR E$=CHR$(243)
 THEN B=0:GOSUB 600
520 IF E$="F" OR E$="S" THEN B=0:GOSUB 850
530 IF E$="N" THEN B=0:GOSUB 1080
540 IF E$="I" THEN B=0:GOSUB 970
550 IF E$="Q" THEN B=0:GOSUB 1400
560 '
570 IF B=1 THEN PRINT CHR$(7);
580 '
590 GOTO 440
600 '
610 'DEPLACER CURSEUR AVEC BUFFER
620 '
630 LOCATE CX,CY
640 PRINT BUFFER$;
650 '
660 'DEPLACER CURSEUR SANS BUFFER
670 '
680 IF E$=CHR$(240) THEN CY=CY-1
690 IF E$=CHR$(241) THEN CY=CY+1
700 IF E$=CHR$(242) THEN CX=CX-1
710 IF E$=CHR$(243) THEN CX=CX+1
720 '
730 IF CX<5  THEN PRINT CHR$(7);:CX=5
740 IF CX>12 THEN PRINT CHR$(7);:CX=12
750 IF CY<9  THEN PRINT CHR$(7);:CY=9
760 IF CY>16 THEN PRINT CHR$(7);:CY=16
770 '
780 LOCATE CX,CY
790 BUFFER$=COPYCHR$(#0) 1
800 '
810 LOCATE CX,CY
820 PRINT CHR$(207);
830 '
840 RETURN
850 '
860 'FIXER POINT - SUPPRIMER POINT
870 '
880 IF FLAG=0 THEN PRINT CHR$(7);:RETURN
890 '
900 LIGNE=7-(CY-9)
910 SUM=2^(8-(CX-4))
```

```
920 '
930 IF E$="F" AND BUFFER$=" " THEN BUFFER$=CHR$(143):CAR(CARACTERE,
LIGNE)=CAR(CARACTERE,LIGNE)+SUM
940 IF E$="S" AND BUFFER$=CHR$(143) THEN BUFFER$=" ":CAR(CARACTERE,
LIGNE)=CAR(CARACTERE,LIGNE)-SUM
950 '
960 RETURN
970 '
980 'INTEGRER
990 '
1000 IF FLAG=0 THEN PRINT CHR$(7);:RETURN
1010 '
1020 SYMBOL CARACTERE,CAR(CARACTERE,7),CAR(CARACTERE,6),CAR(CARACTE
RE,5),CAR(CARACTERE,4),CAR(CARACTERE,3),CAR(CARACTERE,2),CAR(CARACT
ERE,1),CAR(CARACTERE,0)
1030 '
1040 ZZ=ZZ+1
1050 EZ(ZZ)=CARACTERE
1060 '
1070 RETURN
1080 '
1090 'NOUVEAU CARACTERE
1100 '
1110 INPUT#1,"VEUILLEZ CHOISIR UN CARACTERE";CARACTERE
1120 CLS#1
1130 '
1140 IF CARACTERE<0 OR CARACTERE>255 THEN PRINT CHR$(7);:GOTO 1110
1150 '
1160 IF CARACTERE<32 THEN LOCATE 1,1:PRINT CHR$(1);CHR$(CARACTERE);
:GOTO 1190
1170 LOCATE 1,1:PRINT CHR$(CARACTERE);
1180 '
1190 FOR LIGNE=0 TO 7
1200 '
1210 CAR(CARACTERE,LIGNE)=128*TEST(1,385+LIGNE*2)+64*TEST(3,385+LIG
NE*2)+32*TEST(5,385+LIGNE*2)+16*TEST(7,385+LIGNE*2)+8*TEST(9,385+LI
GNE*2)+4*TEST(11,385+LIGNE*2)+2*TEST(13,385+LIGNE*2)+1*TEST(15,385+
LIGNE*2)
1220 '
1230 LOCATE 5,(8-LIGNE)+8
1240 '
1250 FOR I=7 TO 0 STEP -1
```

```
1260 IF (CAR(CARACTERE,LIGNE) AND 2^I)=2^I THEN PRINT CHR$(143); EL
SE PRINT " ";
1270 NEXT I
1280 PRINT
1290 '
1300 NEXT LIGNE
1310 '
1320 LOCATE 1,1
1330 PRINT " ";
1340 '
1350 GOSUB  650
1360 '
1370 FLAG=1
1380 '
1390 RETURN
1400 '
1410 'QUITTER LE PROGRAMME
1420 '
1430 IF ZZ=0 THEN CLS:END
1440 '
1450 PRINT#1,"VOULEZ-VOUS SAUVER LES CARACTERES EDITES SUR
DISQUETTE SOUS FORME D'UN FICHIER PROGRAMME (O/N)"
1460 E$=INKEY$
1470 IF E$="" THEN 1460
1480 '
1490 IF UPPER$(E$)<>"O" THEN CLS:END
1500 '
1510 CLS#1
1520 INPUT#1,"ENTREZ UN NOM DE FICHIER:";FICHIER$
1530 IF LEN(FICHIER$)>8 THEN 1510
1540 OPENOUT FICHIER$
1550 '
1560 PRINT#9,"1 'FICHIER PROGRAMME CREE AUTOMATIQUEMENT"
1570 PRINT#9,"2 SYMBOL AFTER 0"
1580 '
1590 FOR I=1 TO ZZ
1600 PRINT#9,STR$(I*10)+" SYMBOL "+STR$(EZ(I))+","+STR$(CAR(EZ(I),7
))+","+STR$(CAR(EZ(I),6))+","+STR$(CAR(EZ(I),5))+","+STR$(CAR(EZ(I)
,4))+","+STR$(CAR(EZ(I),3))+","+STR$(CAR(EZ(I),2))+","+STR$(CAR(EZ(
I),1))+","+STR$(CAR(EZ(I),0))
1610 NEXT I
1620 '
```

```
1630 CLOSEOUT
1640 '
1650 CLS
1660 END
```

Modification du programme pour le CPC 464 :

```
10 '********
20 DATA &CD,&60,&BB,&32,&7D,&01,&C9,&00
30 FOR A=374 TO 381
40 READ D
50 POKE A,D
60 NEXT A
790 CALL 374:BUFFER$=CHR$(PEEK(381))
```

Description du programme :

100-240 Définitions, dimensionnements et pré-affectations. L'écran est placé en MODE 1, la redéfinition de tous les caractères est rendue possible par SYMBOL AFTER 0) et une fenêtre de dialogue est définie. Cette fenêtre servira au cours du déroulement du programme d'interface d'entrée/sortie entre l'ordinateur et l'utilisateur.

La variable indicée CAR est dimensionnée comme un tableau à deux dimensions de 256 fois 8 éléments. Cette variable doit recevoir les matrices de caractère qui seront éditées pendant le déroulement du programme. Le premier indice du tableau correspond au code du caractère concerné, le second indice étant le numéro de ligne à l'intérieur d'une matrice de caractère, chaque matrice de caractère devant être comprise comme se composant des lignes 7 à 0. Le nombre ainsi appelé sera toujours compris entre 0 et 255. Il reflétera les points fixés et non fixés à l'intérieur de la ligne sélectionnée du caractère voulu.

La variable indicée EZ est dimensionnée comme un tableau de 256 éléments. Elle recevra le numéro des caractères édités. Le dimensionnement à 256

éléments a été choisi pour que chacun des 256 caractères du CPC puisse être redéfini. La variable ZZ contiendra pendant le déroulement du programme l'index de la dernière case occupée dans EZ(x).

Les variables CX et CY qui contiennent la position actuelle du curseur d'édition sont fixées sur leurs valeurs par défaut (coin supérieur gauche de la zone d'édition).

250-320 *Dessiner le cadre de la zone d'édition*
Une ligne est dessinée autour de la surface de huit fois huit caractères qui servira plus tard à l'édition des caractères.

330-410 *Sortir le menu*
La liste des options du programme et les méthodes de sélection de ces possibilités est sortie à côté de la zone d'édition.

430 Le curseur d'édition est placé sur la position par défaut de la zone d'édition (DEPLACER CURSEUR SANS BUFFER).

440-590 *Boucle principale*
La boucle principale constitue le poste d'aiguillage central de l'éditeur de jeu de caractères. C'est à partir de là que sont effectués les sauts aux divers sous-programmes.

Au début de la boucle principale, la variable B est fixée sur la valeur 1. Si cette valeur de B n'a pas été modifiée à la fin de la boucle, c'est qu'une entrée non autorisée a été effectuée et un signal sonore retentit pour signaler une erreur. Une entrée autorisée fixera la valeur de B sur zéro et entraînera un saut au sous-programme voulu (GOSUB). Après retour du sous-programme, la boucle principale sera à nouveau parcourue.

600-640 *Déplacement du curseur avec buffer*
Le fait d'appuyer sur une des quatre touches curseur fera que la boucle principale sautera à ce sous-programme. Dans la position actuelle du curseur sera sorti le contenu de la variable BUFFER$. Le traitement ultérieur est pris en charge par DEPLACER CURSEUR SANS BUFFER.

650-840 *Déplacement du curseur sans buffer*
La direction du mouvement du curseur est d'abord déterminée à partir de la touche curseur enfoncée (E$) et la position du curseur, qui figure dans les variables CX et CY, est recalculée en fonction de cette direction. Si le déplacement voulu débouche sur une position du curseur qui se situerait en dehors de la fenêtre d'édition, un message d'erreur acoustique retentit. La position du curseur est automatiquement ramenée sur la dernière valeur autorisée. Le caractère figurant dans la position de curseur d'édition calculée est lu dans la mémoire écran et sauvé dans la variable BUFFER$ (voir déplacement du curseur avec buffer). Ce n'est que de cette façon qu'on peut rendre possible un déplacement du curseur d'édition sans destruction du contenu de la zone d'édition. Le curseur (CHR$(207)) apparaît alors enfin dans l'emplacement voulu de la zone d'édition. Retour!

850-960 *Fixer point - supprimer point*
En appuyant sur les touches "F" ou "S" on amène la boucle principale à poursuivre ici l'exécution du programme. Si la variable FLAG (=drapeau dans le sens d'indicateur) a la valeur 0, c'est qu'aucun caractère n'a encore été sélectionné pour être édité. Il n'y a donc pas de raison que l'utilisateur tente de fixer ou de supprimer un point; message d'erreur sonore et retour à la boucle principale.

Un indice (LIGNE) est calculé à partir de la position actuelle du curseur (CY). Cet indice permettra l'accès à la variable CAR(x,y) en

fonction de la sortie sur écran. A partir de la position de colonne du curseur (CX), toujours en fonction de la représentation dans la zone d'édition, est calculée la valeur (SUM) du point de la matrice de caractère correspondant à la position du curseur.

La prochaine étape est très simple : si le point dans l'emplacement du curseur doit être fixé (E$="F") et s'il n'était pas fixé auparavant (BUFFER$=""), alors la variable BUFFER$ sera manipulée de façon à laisser derrière elle lors du prochain déplacement du curseur un point d'image fixé (CHR$(143)). Le contenu de la variable CAR, indicée par le code du caractère actuel et par l'indice résultant de la position actuelle du curseur d'édition (LIGNE), est augmenté de la valeur du point correspondant (SUM). Il en va de même pour la suppression d'un point.

970-1070 *Intégration*

Le fait d'actionner la touche "I" a entraîné l'appel de ce sous-programme qui sera abandonné immédiatement, avec un message d'erreur sonore, si aucun caractère n'a encore été sélectionné pour être édité (FLAG=0).

Si un caractère à éditer a été choisi, la variable CARACTERE contient le code du caractère actuellement traité. CARACTERE permet d'accéder à CAR(x,y) et la matrice de caractère actuelle peut être intégrée sous son code de caractère dans le jeu de caractères du CPC (ligne 1020). Cela fait, le contenu de la variable ZZ est augmenté de 1 et EZ(ZZ) reçoit le code du caractère actuel.

1080-1390 *Nouveau caractère*

Le fait d'appuyer sur la touche "N" a fait sauter le programme de la boucle principale au sous-programme NOUVEAU CARACTERE. Le programme attend maintenant que vous entriez un code de caractère à travers la fenêtre de dialogue (#1).

Si l'entrée sort du cadre autorisé, on vous demandera à nouveau d'entrer un code de caractère. Si le contenu de la variable CARACTERE est compris dans l'intervalle autorisé, de 0 à 255, les caractères de commandes qui ne peuvent être représentés directement (0-31) doivent d'abord être isolés (ligne 1160). Le curseur de texte est placé sur la position 1,1 qui permet la sortie graphique de caractères de commande (voir CHR$(1)) et le caractère est sorti dans cet emplacement. La sortie des caractères ordinaires (code de caractère supérieur à 31) est sautée. Si la comparaison des caractères a révélé qu'il ne s'agit pas d'un caractère de commande, le caractère est sorti en ligne 1170.

Dans la boucle FOR-TO-NEXT (lignes 1190-1300), le caractère actuel est lu point par point en position de caractère 1,1, dans la mémoire écran, et la variable CAR est remplie conformément à son indexation. Les lignes 1250 à 1270 ont pour seule fonction de réaliser l'agrandissement du caractère actuel dans la zone d'édition. Après exécution de cette boucle, le caractère est à nouveau effacé du coin supérieur gauche; il se trouve désormais aussi bien dans la variable CAR(x,y) que, sous une forme agrandie, sur l'écran.

Avant le retour à la boucle principale, le curseur est encore placé dans la zone d'édition, sans sortir BUFFER$, et la variable FLAG est fixée sur la valeur 1, ce qui indique qu'un caractère est disponible.

1400-Fin *Fin du programme*
Si la touche "Q" a été actionnée à l'intérieur de la boucle principale, on saute au sous-programme QUITTER LE PROGRAMME (Attention, il n'est pas possible de revenir à la boucle principale une fois qu'on se trouve dans cette partie du programme!).

La première possibilité pour que le programme prenne fin est que la variable ZZ vaille 0, c'est-à-dire qu'aucun nouveau caractère n'ait été intégré dans le jeu de caractères.

La seconde possibilité de fin du programme est une réponse autre que "O" à la question de savoir si les caractères édités doivent être sauvegardés sur disquette sous forme d'un fichier de programme. Si vous avez répondu en appuyant sur la touche "O", l'ordinateur vous demande dans la fenêtre de dialogue quel nom il doit donner au fichier de programme à créer. Une fois le nom correctement entré, un fichier est ouvert en sortie, sous ce nom, sur la disquette.

Dans ce fichier est tout d'abord écrit le commentaire indiquant qu'il s'agit d'un fichier de programme créé automatiquement. On y écrit ensuite l'instruction SYMBOL AFTER 0 qui autorise une redéfinition du jeu de caractères tout entier. Après cela, tous les caractères redéfinis sont écrits dans le fichier de sortie sous forme de lignes de programme BASIC complètes, avec des numéros de ligne croissants, avec l'instruction SYMBOL, avec les codes de caractères et les matrices de caractère correspondantes. Le jeu combiné de la variable ZZ et de la variable indicée EZ permet de s'assurer que ne soient sauvegardés sur disquette que les caractères qui ont réellement été redéfinis. Tous les autres sont donc ignorés ici.

La troisième et dernière possibilité de fin du programme est donc que les caractères édités aient été sauvegardés sur disquette sous la forme d'un fichier de programme.

3.5 LES ACCENTS POUR LE CPC

Il vous semble certainement, comme à nous, insupportable de devoir se passer des accents lorsqu'on travaille sur le CPC. L'absence d'accents et le remplacement de ç par c dans des mots français paraîssent toujours artificiels et étranges sans parler de certains cas où les accents aident à lire tel ou tel mot ou à éviter une confusion possible. C'est pourquoi nous avons réalisé les matrices de caractère pour les lettres à, é, è, ù et ç en nous basant sur la forme des caractères déjà existants pour que les lettres accentuées s'harmonisent bien avec le reste du jeu de caractères.

Comme les accents sont d'un emploi fréquent en français, il est souhaitable qu'on puisse les taper sur l'ordinateur directement au clavier, exactement comme les autres caractères, sans avoir à jongler péniblement avec les codes CHR$. Nous avons donc redéfini les touches suivantes :

Code caractère	Ancien caractère	Nouveau caractère
64	@	à
91	[	°
92	\	ç
93	]	§
123	{	é
124	|	ù
125	}	è

Nous reconnaissons bien volontiers qu'un clavier ainsi redéfini a quelque chose de curieux et de paradoxal puisqu'il s'agit bien en réalité d'un clavier américain (QWERTY) avec des accents français. Nous avons cependant renoncé à redéfinir l'emplacement des touches non accentuées pour éviter tout risque de confusion entre l'affectation des touches normales en BASIC et celles qui seraient utilisées uniquement pour le traitement de texte. Si vous préférez toutefois disposer d'un véritable clavier français (AZERTY), vous pouvez redéfinir l'emplacement des touches à tout moment avec KEY DEF.

```
1000 '*********************************************************
1001 '***                                                   ***
1002 '***         ACCENTS POUR LE CPC  JST 2.6.1986         ***
1003 '***                                                   ***
1004 '*********************************************************
1005 '
1006 SYMBOL AFTER 63
1007 '
1008 FOR I=0 TO 6
1009 '
1010 READ CH
1011 '
1012 FOR J=0 TO 7
1013 READ X
1014 TX(J)=X
1015 NEXT J
1016 '
1017 SYMBOL CH,TX(0),TX(1),TX(2),TX(3),TX(4),TX(5),TX(6),TX(7)
1018 '
1019 NEXT I
1020 '
1021 NEW
1022 '
1023 DATA 64
1024 '
1025 DATA &X01100000        ' ■■
1026 DATA &X00110000        '  ■■
1027 DATA &X01111000        ' ■■■■
1028 DATA &X00001100        '    ■■
1029 DATA &X01111100        ' ■■■■■
1030 DATA &X11001100        '■■  ■■
1031 DATA &X01110110        ' ■■■ ■■
1032 DATA &X00000000        '
1033 '
1034 DATA 91
1035 '
1036 DATA &X00011000        '   ■■
1037 DATA &X00100100        '  ■  ■
1038 DATA &X00100100        '  ■  ■
1039 DATA &X00011000        '   ■■
```

```
1040 DATA &X00000000        '
1041 DATA &X00000000        '
1042 DATA &X00000000        '
1043 DATA &X00000000        '
1044 '
1045 DATA 92
1046 '
1047 DATA &X00000000        '
1048 DATA &X00000000        '
1049 DATA &X00111100        '  ■■■■
1050 DATA &X01100110        ' ■■  ■■
1051 DATA &X01100000        ' ■■
1052 DATA &X01100110        ' ■■  ■■
1053 DATA &X00111100        '  ■■■■
1054 DATA &X00011000        '   ■■
1055 '
1056 DATA 93
1057 '
1058 DATA &X00011110        '   ■■■■
1059 DATA &X00110000        '  ■■
1060 DATA &X00111000        '  ■■■
1061 DATA &X01101100        ' ■■ ■■
1062 DATA &X00111000        '  ■■■
1063 DATA &X00011000        '   ■■
1064 DATA &X11110000        '■■■■
1065 DATA &X00000000        '
1066 '
1067 DATA 123
1068 '
1069 DATA &X00001100        '    ■■
1070 DATA &X00011000        '   ■■
1071 DATA &X00111100        '  ■■■■
1072 DATA &X01100110        ' ■■  ■■
1073 DATA &X01111110        ' ■■■■■■
1074 DATA &X01100000        ' ■■
1075 DATA &X00111100        '  ■■■■
1076 DATA &X00000000        '
1077 '
1078 DATA 124
1079 '
1080 DATA &X00110000        '  ■■
1081 DATA &X00011000        '   ■■
1082 DATA &X01100110        ' ■■  ■■
```

```
1083 DATA &X01100110      ' ▮▮  ▮▮
1084 DATA &X01100110      ' ▮▮  ▮▮
1085 DATA &X01100110      ' ▮▮  ▮▮
1086 DATA &X00111111      '  ▮▮▮▮▮▮
1087 DATA &X00000000      '
1088 '
1089 DATA 125
1090 '
1091 DATA &X00110000      '  ▮▮
1092 DATA &X00011000      '   ▮▮
1093 DATA &X00111100      '  ▮▮▮▮
1094 DATA &X01100110      ' ▮▮  ▮▮
1095 DATA &X01111110      ' ▮▮▮▮▮▮
1096 DATA &X01100000      ' ▮▮
1097 DATA &X00111100      '  ▮▮▮▮
1098 DATA &X00000000      '
```

Description du programme :

L'en-tête du programme est suivi en ligne 1006 par l'instruction BASIC SYMBOL AFTER 64 qui permet la redéfinition des caractères à partir du caractère de code 64. Les lignes 1008 à 1019 contiennent une boucle FOR-TO-NEXT qui intègre les caractères "à°ç§ éùè" dans le jeu de caractères. Si vous l'examinez de plus près, vous verrez que cette boucle lit d'abord une instruction DATA qui contient le code de caractère. Dans une seconde boucle FOR-TO-NEXT sont ensuite lues, toujours dans des instructions DATA, les huit valeurs numériques contenant la matrice de caractère; ces valeurs sont affectées aux variables indicées TX(0) à TX(7). La matrice est enfin intégrée dans le jeu de caractères sous le code de caractère actuel (CH) (ligne 1017). On procède de même pour les sept caractères. Une fois la redéfinition terminée, le programme est vidé de la mémoire (ligne 1021)!

De la ligne 1022 à la fin du programme figurent des instructions DATA par groupes de neuf. Le premier DATA de chacun de ces groupes contient le numéro du caractère qui doit être redéfini. Les huit DATAs suivants contiennent la matrice de caractère. Pour les sept groupes de neuf, les matrices de caractère ont été imprimées en écriture binaire pour que vous puissiez plus aisément faire le rapprochement entre les valeurs numériques et les formes de caractère qu'elles représentent.

Les commentaires que vous trouverez dans les lignes d'instructions DATA répondent également à ce même souci de concret et ils ne doivent pas être tapés dans votre programme.

3.6 UN JEU DE CARACTERES ALTERNATIF POUR LE CPC

Les CPCs se distinguent notamment par le fait qu'ils peuvent représenter sur l'écran 80 caractères par ligne. Ce simple fait donne à ces ordinateurs un aspect de professionnalité qui est cependant atténué par le fait que les caractères du jeu de caractères intégré sont un peu fatigants à lire en MODE 2. Si l'on utilise son CPC pour des applications professionnelles, comme par exemple le traitement de texte ou la programmation, on aspire vite à pouvoir disposer d'un jeu de caractères semblable à ceux utilisés sur les bons terminaux de texte.

Il s'agit en effet de jeux de caractères qui frappent surtout par leur simplicité, les traits verticaux ou horizontaux sont ainsi en général d'une largeur d'un point simple. Les éléments simples de ligne peuvent cependant causer des problèmes d'affichage, surtout sur les moniteurs couleur, mais l'expérience révèle que ces problèmes dépendent beaucoup de la combinaison de couleurs utilisée.

Pour écrire tous les programmes nécessitant un temps de travail important, nous avons utilisé systématiquement le jeu de caractères que nous allons maintenant vous proposer. Nous avons constaté ce faisant que la combinaison de couleurs la plus lisible en mode 80 colonnes est la suivante :

```
BORDER 0
INK 0,0
INK 1,11
```

Si vous souhaitez bénéficier vous aussi de notre jeu de caractères alternatif dans votre travail, vous avez le choix soit de taper le programme dont voici le listing, soit de reproduire à l'aide de l'éditeur du jeu de caractères les formes de caractères que nous avons fait imprimer à cet effet sous forme de commentaires à côté des matrices.

Notez bien d'ailleurs que les commentaires placés dans les instructions DATA ont simplement pour but de vous faciliter la lecture du programme et qu'il ne faut pas les taper.

```
1000 '*********************************************************
1001 '***                                                   ***
1002 '***     UN JEU DE CARACTERES ALTERNATIF POUR LE CPC   ***
1003 '***                  JST/TAV 2.6.1986                 ***
1004 '***                                                   ***
1005 '*********************************************************
1006 '
1007 SYMBOL AFTER 33
1008 '
1009 FOR I=33 TO 125
1010 '
1011 FOR J=0 TO 7
1012 READ X
1013 TX(J)=X
1014 NEXT J
1015 '
1016 SYMBOL I,TX(0),TX(1),TX(2),TX(3),TX(4),TX(5),TX(6),TX(7)
1017 '
1018 NEXT I
1019 '
1020 NEW
1021 '
1022 '!
1023 '
1024 DATA &X00001000       '    ▮
1025 DATA &X00001000       '    ▮
1026 DATA &X00001000       '    ▮
1027 DATA &X00001000       '    ▮
1028 DATA &X00001000       '    ▮
1029 DATA &X00000000       '
1030 DATA &X00001000       '    ▮
1031 DATA &X00000000       '
1032 '
1033 '"
1034 '
1035 DATA &X00100100       '  ▮  ▮
1036 DATA &X00100100       '  ▮  ▮
1037 DATA &X00100100       '  ▮  ▮
```

```
1038 DATA &X00000000         '
1039 DATA &X00000000         '
1040 DATA &X00000000         '
1041 DATA &X00000000         '
1042 DATA &X00000000         '
1043 '
1044 '#
1045 '
1046 DATA &X00100100         '  ▮  ▮
1047 DATA &X00100100         '  ▮  ▮
1048 DATA &X01111110         ' ▮▮▮▮▮▮
1049 DATA &X00100100         '  ▮  ▮
1050 DATA &X01111110         ' ▮▮▮▮▮▮
1051 DATA &X00100100         '  ▮  ▮
1052 DATA &X00100100         '  ▮  ▮
1053 DATA &X00000000         '
1054 '
1055 '$
1056 '
1057 DATA &X00001000         '    ▮
1058 DATA &X00011110         '   ▮▮▮▮
1059 DATA &X00100000         '  ▮
1060 DATA &X00011100         '   ▮▮▮
1061 DATA &X00000010         '      ▮
1062 DATA &X00111100         '  ▮▮▮▮
1063 DATA &X00001000         '    ▮
1064 DATA &X00000000         '
1065 '
1066 '%
1067 '
1068 DATA &X00000000         '
1069 DATA &X01100010         ' ▮▮   ▮
1070 DATA &X01100100         ' ▮▮  ▮
1071 DATA &X00001000         '    ▮
1072 DATA &X00010000         '   ▮
1073 DATA &X00100110         '  ▮  ▮▮
1074 DATA &X01000110         ' ▮   ▮▮
1075 DATA &X00000000         '
1076 '
1077 '&
1078 '
1079 DATA &X00110000         '  ▮▮
```

```
1080 DATA &X01001000       ' █  █
1081 DATA &X01001000       ' █  █
1082 DATA &X00110000       '  ██
1083 DATA &X01001010       ' █  █ █
1084 DATA &X01000100       ' █   █
1085 DATA &X00111010       '  ███ █
1086 DATA &X00000000       '
1087 '
1088 ''
1089 '
1090 DATA &X00001000       '    █
1091 DATA &X00010000       '   █
1092 DATA &X00100000       '  █
1093 DATA &X00000000       '
1094 DATA &X00000000       '
1095 DATA &X00000000       '
1096 DATA &X00000000       '
1097 DATA &X00000000       '
1098 '
1099 '(
1100 '
1101 DATA &X00000100       '     █
1102 DATA &X00001000       '    █
1103 DATA &X00010000       '   █
1104 DATA &X00010000       '   █
1105 DATA &X00010000       '   █
1106 DATA &X00001000       '    █
1107 DATA &X00000100       '     █
1108 DATA &X00000000       '
1109 '
1110 ')
1111 '
1112 DATA &X00100000       '  █
1113 DATA &X00010000       '   █
1114 DATA &X00001000       '    █
1115 DATA &X00001000       '    █
1116 DATA &X00001000       '    █
1117 DATA &X00010000       '   █
1118 DATA &X00100000       '  █
1119 DATA &X00000000       '
1120 '
1121 '*
```

```
1122 '
1123 DATA &X00001000        '    ■
1124 DATA &X00101010        '  ■ ■ ■
1125 DATA &X00011100        '   ■■■
1126 DATA &X00111110        '  ■■■■■
1127 DATA &X00011100        '   ■■■
1128 DATA &X00101010        '  ■ ■ ■
1129 DATA &X00001000        '    ■
1130 DATA &X00000000        '
1131 '
1132 '+
1133 '
1134 DATA &X00000000        '
1135 DATA &X00001000        '    ■
1136 DATA &X00001000        '    ■
1137 DATA &X00111110        '  ■■■■■
1138 DATA &X00001000        '    ■
1139 DATA &X00001000        '    ■
1140 DATA &X00000000        '
1141 DATA &X00000000        '
1142 '
1143 ',
1144 '
1145 DATA &X00000000        '
1146 DATA &X00000000        '
1147 DATA &X00000000        '
1148 DATA &X00000000        '
1149 DATA &X00000000        '
1150 DATA &X00001000        '    ■
1151 DATA &X00001000        '    ■
1152 DATA &X00010000        '   ■
1153 '
1154 '-
1155 '
1156 DATA &X00000000        '
1157 DATA &X00000000        '
1158 DATA &X00000000        '
1159 DATA &X01111110        ' ■■■■■■
1160 DATA &X00000000        '
1161 DATA &X00000000        '
1162 DATA &X00000000        '
1163 DATA &X00000000        '
```

```
1164 '
1165 '.
1166 '
1167 DATA &X00000000       '
1168 DATA &X00000000       '
1169 DATA &X00000000       '
1170 DATA &X00000000       '
1171 DATA &X00000000       '
1172 DATA &X00000000       '
1173 DATA &X00001000       '    ▮
1174 DATA &X00000000       '
1175 '
1176 '/
1177 '
1178 DATA &X00000000       '
1179 DATA &X00000010       '      ▮
1180 DATA &X00000100       '     ▮
1181 DATA &X00001000       '    ▮
1182 DATA &X00010000       '   ▮
1183 DATA &X00100000       '  ▮
1184 DATA &X01000000       ' ▮
1185 DATA &X00000000       '
1186 '
1187 '0
1188 '
1189 DATA &X00111000       '  ▮▮▮
1190 DATA &X01000100       ' ▮   ▮
1191 DATA &X01001100       ' ▮  ▮▮
1192 DATA &X01010100       ' ▮ ▮ ▮
1193 DATA &X01100100       ' ▮▮  ▮
1194 DATA &X01000100       ' ▮   ▮
1195 DATA &X00111000       '  ▮▮▮
1196 DATA &X00000000       '
1197 '
1198 '1
1199 '
1200 DATA &X00010000       '   ▮
1201 DATA &X00110000       '  ▮▮
1202 DATA &X01010000       ' ▮ ▮
1203 DATA &X00010000       '   ▮
1204 DATA &X00010000       '   ▮
1205 DATA &X00010000       '   ▮
```

```
1206 DATA &X01111100        ' ▮▮▮▮▮
1207 DATA &X00000000        '
1208 '
1209 '2
1210 '
1211 DATA &X00111100        '  ▮▮▮▮
1212 DATA &X01000010        ' ▮    ▮
1213 DATA &X00000010        '      ▮
1214 DATA &X00001100        '    ▮▮
1215 DATA &X00110000        '  ▮▮
1216 DATA &X01000000        ' ▮
1217 DATA &X01111110        ' ▮▮▮▮▮▮
1218 DATA &X00000000        '
1219 '
1220 '3
1221 '
1222 DATA &X00111100        '  ▮▮▮▮
1223 DATA &X01000010        ' ▮    ▮
1224 DATA &X00000010        '      ▮
1225 DATA &X00011100        '   ▮▮▮
1226 DATA &X00000010        '      ▮
1227 DATA &X01000010        ' ▮    ▮
1228 DATA &X00111100        '  ▮▮▮▮
1229 DATA &X00000000        '
1230 '
1231 '4
1232 '
1233 DATA &X00000100        '     ▮
1234 DATA &X00001100        '    ▮▮
1235 DATA &X00010100        '   ▮ ▮
1236 DATA &X00100100        '  ▮  ▮
1237 DATA &X01111110        ' ▮▮▮▮▮▮
1238 DATA &X00000100        '     ▮
1239 DATA &X00000100        '     ▮
1240 DATA &X00000000        '
1241 '
1242 '5
1243 '
1244 DATA &X01111110        ' ▮▮▮▮▮▮
1245 DATA &X01000000        ' ▮
1246 DATA &X01111000        ' ▮▮▮▮
1247 DATA &X00000100        '     ▮
```

```
1248 DATA &X00000010       '      ▮
1249 DATA &X01000100       ' ▮   ▮
1250 DATA &X00111000       '  ▮▮▮
1251 DATA &X00000000       '
1252 '
1253 '6
1254 '
1255 DATA &X00011100       '   ▮▮▮
1256 DATA &X00100000       '  ▮
1257 DATA &X01000000       ' ▮
1258 DATA &X01111100       ' ▮▮▮▮▮
1259 DATA &X01000010       ' ▮    ▮
1260 DATA &X01000010       ' ▮    ▮
1261 DATA &X00111100       '  ▮▮▮▮
1262 DATA &X00000000       '
1263 '
1264 '7
1265 '
1266 DATA &X01111110       ' ▮▮▮▮▮▮
1267 DATA &X01000010       ' ▮    ▮
1268 DATA &X00000100       '     ▮
1269 DATA &X00001000       '    ▮
1270 DATA &X00001000       '    ▮
1271 DATA &X00001000       '    ▮
1272 DATA &X00001000       '    ▮
1273 DATA &X00000000       '
1274 '
1275 '8
1276 '
1277 DATA &X00111100       '  ▮▮▮▮
1278 DATA &X01000010       ' ▮    ▮
1279 DATA &X01000010       ' ▮    ▮
1280 DATA &X00111100       '  ▮▮▮▮
1281 DATA &X01000010       ' ▮    ▮
1282 DATA &X01000010       ' ▮    ▮
1283 DATA &X00111100       '  ▮▮▮▮
1284 DATA &X00000000       '
1285 '
1286 '9
1287 '
1288 DATA &X00111100       '  ▮▮▮▮
1289 DATA &X01000010       ' ▮    ▮
```

```
1290 DATA &X01000010      ' █    █
1291 DATA &X00111110      '  █████
1292 DATA &X00000010      '      █
1293 DATA &X00000100      '     █
1294 DATA &X00111000      '  ███
1295 DATA &X00000000      '
1296 '
1297 ':
1298 '
1299 DATA &X00000000      '
1300 DATA &X00000000      '
1301 DATA &X00001000      '    █
1302 DATA &X00000000      '
1303 DATA &X00000000      '
1304 DATA &X00001000      '    █
1305 DATA &X00000000      '
1306 DATA &X00000000      '
1307 '
1308 ';
1309 '
1310 DATA &X00000000      '
1311 DATA &X00000000      '
1312 DATA &X00001000      '    █
1313 DATA &X00000000      '
1314 DATA &X00000000      '
1315 DATA &X00001000      '    █
1316 DATA &X00001000      '    █
1317 DATA &X00010000      '   █
1318 '
1319 '<
1320 '
1321 DATA &X00000100      '     █
1322 DATA &X00001000      '    █
1323 DATA &X00010000      '   █
1324 DATA &X00100000      '  █
1325 DATA &X00010000      '   █
1326 DATA &X00001000      '    █
1327 DATA &X00000100      '     █
1328 DATA &X00000000      '
1329 '
1330 '=
1331 '
```

```
1332 DATA &X00000000      '
1333 DATA &X00000000      '
1334 DATA &X01111110      ' ▮▮▮▮▮▮
1335 DATA &X00000000      '
1336 DATA &X01111110      ' ▮▮▮▮▮▮
1337 DATA &X00000000      '
1338 DATA &X00000000      '
1339 DATA &X00000000      '
1340 '
1341 '>
1342 '
1343 DATA &X00100000      '  ▮
1344 DATA &X00010000      '   ▮
1345 DATA &X00001000      '    ▮
1346 DATA &X00000100      '     ▮
1347 DATA &X00001000      '    ▮
1348 DATA &X00010000      '   ▮
1349 DATA &X00100000      '  ▮
1350 DATA &X00000000      '
1351 '
1352 '?
1353 '
1354 DATA &X00111100      '  ▮▮▮▮
1355 DATA &X01000010      ' ▮    ▮
1356 DATA &X00000010      '      ▮
1357 DATA &X00001100      '    ▮▮
1358 DATA &X00010000      '   ▮
1359 DATA &X00000000      '
1360 DATA &X00010000      '   ▮
1361 DATA &X00000000      '
1362 '
1363 '@
1364 '
1365 DATA &X00011100      '   ▮▮▮
1366 DATA &X00100010      '  ▮   ▮
1367 DATA &X01001010      ' ▮  ▮ ▮
1368 DATA &X01010110      ' ▮ ▮ ▮▮
1369 DATA &X01001100      ' ▮  ▮▮
1370 DATA &X00100000      '  ▮
1371 DATA &X00011100      '   ▮▮▮
1372 DATA &X00000000      '
1373 '
```

```
1374 'A
1375 '
1376 DATA &X00011000         '   ■■
1377 DATA &X00100100         '  ■  ■
1378 DATA &X01000010         ' ■    ■
1379 DATA &X01111110         ' ■■■■■■
1380 DATA &X01000010         ' ■    ■
1381 DATA &X01000010         ' ■    ■
1382 DATA &X01000010         ' ■    ■
1383 DATA &X00000000         '
1384 '
1385 'B
1386 '
1387 DATA &X01111100         ' ■■■■■
1388 DATA &X00100010         '  ■   ■
1389 DATA &X00100010         '  ■   ■
1390 DATA &X00111100         '  ■■■■
1391 DATA &X00100010         '  ■   ■
1392 DATA &X00100010         '  ■   ■
1393 DATA &X01111100         ' ■■■■■
1394 DATA &X00000000         '
1395 '
1396 'C
1397 '
1398 DATA &X00011100         '   ■■■
1399 DATA &X00100010         '  ■   ■
1400 DATA &X01000000         ' ■
1401 DATA &X01000000         ' ■
1402 DATA &X01000000         ' ■
1403 DATA &X00100010         '  ■   ■
1404 DATA &X00011100         '   ■■■
1405 DATA &X00000000         '
1406 '
1407 'D
1408 '
1409 DATA &X01111000         ' ■■■■
1410 DATA &X00100100         '  ■  ■
1411 DATA &X00100010         '  ■   ■
1412 DATA &X00100010         '  ■   ■
1413 DATA &X00100010         '  ■   ■
1414 DATA &X00100100         '  ■  ■
1415 DATA &X01111000         ' ■■■■
```

```
1416 DATA &X00000000        '
1417 '
1418 'E
1419 '
1420 DATA &X01111110        ' ▮▮▮▮▮▮
1421 DATA &X01000000        ' ▮
1422 DATA &X01000000        ' ▮
1423 DATA &X01111100        ' ▮▮▮▮▮
1424 DATA &X01000000        ' ▮
1425 DATA &X01000000        ' ▮
1426 DATA &X01111110        ' ▮▮▮▮▮▮
1427 DATA &X00000000        '
1428 '
1429 'F
1430 '
1431 DATA &X01111110        ' ▮▮▮▮▮▮
1432 DATA &X01000000        ' ▮
1433 DATA &X01000000        ' ▮
1434 DATA &X01111100        ' ▮▮▮▮▮
1435 DATA &X01000000        ' ▮
1436 DATA &X01000000        ' ▮
1437 DATA &X01000000        ' ▮
1438 DATA &X00000000        '
1439 '
1440 'G
1441 '
1442 DATA &X00011100        '   ▮▮▮
1443 DATA &X00100010        '  ▮   ▮
1444 DATA &X01000000        ' ▮
1445 DATA &X01001110        ' ▮  ▮▮▮
1446 DATA &X01000010        ' ▮    ▮
1447 DATA &X00100010        '  ▮   ▮
1448 DATA &X00011100        '   ▮▮▮
1449 DATA &X00000000        '
1450 '
1451 'H
1452 '
1453 DATA &X01000010        ' ▮    ▮
1454 DATA &X01000010        ' ▮    ▮
1455 DATA &X01000010        ' ▮    ▮
1456 DATA &X01111110        ' ▮▮▮▮▮▮
1457 DATA &X01000010        ' ▮    ▮
```

```
1458 DATA &X01000010        ' ▮    ▮
1459 DATA &X01000010        ' ▮    ▮
1460 DATA &X00000000        '
1461 '
1462 'I
1463 '
1464 DATA &X00011100        '   ▮▮▮
1465 DATA &X00001000        '    ▮
1466 DATA &X00001000        '    ▮
1467 DATA &X00001000        '    ▮
1468 DATA &X00001000        '    ▮
1469 DATA &X00001000        '    ▮
1470 DATA &X00011100        '   ▮▮▮
1471 DATA &X00000000        '
1472 '
1473 'J
1474 '
1475 DATA &X00001110        '    ▮▮▮
1476 DATA &X00000100        '     ▮
1477 DATA &X00000100        '     ▮
1478 DATA &X00000100        '     ▮
1479 DATA &X00000100        '     ▮
1480 DATA &X01000100        ' ▮   ▮
1481 DATA &X00111000        '  ▮▮▮
1482 DATA &X00000000        '
1483 '
1484 'K
1485 '
1486 DATA &X01000010        ' ▮    ▮
1487 DATA &X01000100        ' ▮   ▮
1488 DATA &X01001000        ' ▮  ▮
1489 DATA &X01110000        ' ▮▮▮
1490 DATA &X01001000        ' ▮  ▮
1491 DATA &X01000100        ' ▮   ▮
1492 DATA &X01000010        ' ▮    ▮
1493 DATA &X00000000        '
1494 '
1495 'L
1496 '
1497 DATA &X01000000        ' ▮
1498 DATA &X01000000        ' ▮
1499 DATA &X01000000        ' ▮
```

```
1500 DATA &X01000000        ' ▮
1501 DATA &X01000000        ' ▮
1502 DATA &X01000000        ' ▮
1503 DATA &X01111110        ' ▮▮▮▮▮▮
1504 DATA &X00000000        '
1505 '
1506 'M
1507 '
1508 DATA &X01000010        ' ▮    ▮
1509 DATA &X01100110        ' ▮▮  ▮▮
1510 DATA &X01011010        ' ▮ ▮▮ ▮
1511 DATA &X01011010        ' ▮ ▮▮ ▮
1512 DATA &X01000010        ' ▮    ▮
1513 DATA &X01000010        ' ▮    ▮
1514 DATA &X01000010        ' ▮    ▮
1515 DATA &X00000000        '
1516 '
1517 'N
1518 '
1519 DATA &X01000010        ' ▮    ▮
1520 DATA &X01100010        ' ▮▮   ▮
1521 DATA &X01010010        ' ▮ ▮  ▮
1522 DATA &X01001010        ' ▮  ▮ ▮
1523 DATA &X01000110        ' ▮   ▮▮
1524 DATA &X01000010        ' ▮    ▮
1525 DATA &X01000010        ' ▮    ▮
1526 DATA &X00000000        '
1527 '
1528 'O
1529 '
1530 DATA &X00111100        '  ▮▮▮▮
1531 DATA &X01000010        ' ▮    ▮
1532 DATA &X01000010        ' ▮    ▮
1533 DATA &X01000010        ' ▮    ▮
1534 DATA &X01000010        ' ▮    ▮
1535 DATA &X01000010        ' ▮    ▮
1536 DATA &X00111100        '  ▮▮▮▮
1537 DATA &X00000000        '
1538 '
1539 'P
1540 '
1541 DATA &X01111100        ' ▮▮▮▮▮
```

```
1542 DATA &X01000010       ' ▮    ▮
1543 DATA &X01000010       ' ▮    ▮
1544 DATA &X01111100       ' ▮▮▮▮▮
1545 DATA &X01000000       ' ▮
1546 DATA &X01000000       ' ▮
1547 DATA &X01000000       ' ▮
1548 DATA &X00000000       '
1549 '
1550 'Q
1551 '
1552 DATA &X00111100       '  ▮▮▮▮
1553 DATA &X01000010       ' ▮    ▮
1554 DATA &X01000010       ' ▮    ▮
1555 DATA &X01000010       ' ▮    ▮
1556 DATA &X01001010       ' ▮  ▮ ▮
1557 DATA &X01000100       ' ▮   ▮
1558 DATA &X00111010       '  ▮▮▮ ▮
1559 DATA &X00000000       '
1560 '
1561 'R
1562 '
1563 DATA &X01111100       ' ▮▮▮▮▮
1564 DATA &X01000010       ' ▮    ▮
1565 DATA &X01000010       ' ▮    ▮
1566 DATA &X01111100       ' ▮▮▮▮▮
1567 DATA &X01001000       ' ▮  ▮
1568 DATA &X01000100       ' ▮   ▮
1569 DATA &X01000010       ' ▮    ▮
1570 DATA &X00000000       '
1571 '
1572 'S
1573 '
1574 DATA &X00111100       '  ▮▮▮▮
1575 DATA &X01000010       ' ▮    ▮
1576 DATA &X01000000       ' ▮
1577 DATA &X00111100       '  ▮▮▮▮
1578 DATA &X00000010       '      ▮
1579 DATA &X01000010       ' ▮    ▮
1580 DATA &X00111100       '  ▮▮▮▮
1581 DATA &X00000000       '
1582 '
1583 'T
```

```
1584 '
1585 DATA &X00111110        '  █████
1586 DATA &X00001000        '    █
1587 DATA &X00001000        '    █
1588 DATA &X00001000        '    █
1589 DATA &X00001000        '    █
1590 DATA &X00001000        '    █
1591 DATA &X00001000        '    █
1592 DATA &X00000000        '
1593 '
1594 'U
1595 '
1596 DATA &X01000010        ' █    █
1597 DATA &X01000010        ' █    █
1598 DATA &X01000010        ' █    █
1599 DATA &X01000010        ' █    █
1600 DATA &X01000010        ' █    █
1601 DATA &X01000010        ' █    █
1602 DATA &X00111100        '  ████
1603 DATA &X00000000        '
1604 '
1605 'V
1606 '
1607 DATA &X01000010        ' █    █
1608 DATA &X01000010        ' █    █
1609 DATA &X01000010        ' █    █
1610 DATA &X00100100        '  █  █
1611 DATA &X00100100        '  █  █
1612 DATA &X00011000        '   ██
1613 DATA &X00011000        '   ██
1614 DATA &X00000000        '
1615 '
1616 'W
1617 '
1618 DATA &X01000010        ' █    █
1619 DATA &X01000010        ' █    █
1620 DATA &X01000010        ' █    █
1621 DATA &X01011010        ' █ ██ █
1622 DATA &X01011010        ' █ ██ █
1623 DATA &X01100110        ' ██  ██
1624 DATA &X01000010        ' █    █
1625 DATA &X00000000        '
```

```
1626 '
1627 'X
1628 '
1629 DATA &X01000010        ' ■    ■
1630 DATA &X01000010        ' ■    ■
1631 DATA &X00100100        '  ■  ■
1632 DATA &X00011000        '   ■■
1633 DATA &X00100100        '  ■  ■
1634 DATA &X01000010        ' ■    ■
1635 DATA &X01000010        ' ■    ■
1636 DATA &X00000000        '
1637 '
1638 'Y
1639 '
1640 DATA &X00100010        '  ■   ■
1641 DATA &X00100010        '  ■   ■
1642 DATA &X00100010        '  ■   ■
1643 DATA &X00011100        '   ■■■
1644 DATA &X00001000        '    ■
1645 DATA &X00001000        '    ■
1646 DATA &X00001000        '    ■
1647 DATA &X00000000        '
1648 '
1649 'Z
1650 '
1651 DATA &X01111110        ' ■■■■■■
1652 DATA &X00000010        '      ■
1653 DATA &X00000100        '     ■
1654 DATA &X00011000        '   ■■
1655 DATA &X00100000        '  ■
1656 DATA &X01000000        ' ■
1657 DATA &X01111110        ' ■■■■■■
1658 DATA &X00000000        '
1659 '
1660 '[
1661 '
1662 DATA &X00011100        '   ■■■
1663 DATA &X00010000        '   ■
1664 DATA &X00010000        '   ■
1665 DATA &X00010000        '   ■
1666 DATA &X00010000        '   ■
1667 DATA &X00010000        '   ■
```

```
1668 DATA &X00011100        '   ▮▮▮
1669 DATA &X00000000        '
1670 '
1671 '\
1672 '
1673 DATA &X00000000        '
1674 DATA &X01000000        ' ▮
1675 DATA &X00100000        '  ▮
1676 DATA &X00010000        '   ▮
1677 DATA &X00001000        '    ▮
1678 DATA &X00000100        '     ▮
1679 DATA &X00000010        '      ▮
1680 DATA &X00000000        '
1681 '
1682 ']
1683 '
1684 DATA &X00111000        '  ▮▮▮
1685 DATA &X00001000        '    ▮
1686 DATA &X00001000        '    ▮
1687 DATA &X00001000        '    ▮
1688 DATA &X00001000        '    ▮
1689 DATA &X00001000        '    ▮
1690 DATA &X00111000        '  ▮▮▮
1691 DATA &X00000000        '
1692 '
1693 '^
1694 '
1695 DATA &X00001000        '    ▮
1696 DATA &X00011100        '   ▮▮▮
1697 DATA &X00101010        '  ▮ ▮ ▮
1698 DATA &X00001000        '    ▮
1699 DATA &X00001000        '    ▮
1700 DATA &X00001000        '    ▮
1701 DATA &X00001000        '    ▮
1702 DATA &X00000000        '
1703 '
1704 '_
1705 '
1706 DATA &X00000000        '
1707 DATA &X00000000        '
1708 DATA &X00000000        '
1709 DATA &X00000000        '
```

```
1710 DATA &X00000000      '
1711 DATA &X00000000      '
1712 DATA &X00000000      '
1713 DATA &X11111111      '████████
1714 '
1715 '`
1716 '
1717 DATA &X00010000      '   █
1718 DATA &X00001000      '    █
1719 DATA &X00000100      '     █
1720 DATA &X00000000      '
1721 DATA &X00000000      '
1722 DATA &X00000000      '
1723 DATA &X00000000      '
1724 DATA &X00000000      '
1725 '
1726 'a
1727 '
1728 DATA &X00000000      '
1729 DATA &X00000000      '
1730 DATA &X00111000      '  ███
1731 DATA &X00000100      '     █
1732 DATA &X00111100      '  ████
1733 DATA &X01000100      ' █   █
1734 DATA &X00111010      '  ███ █
1735 DATA &X00000000      '
1736 '
1737 'b
1738 '
1739 DATA &X01000000      ' █
1740 DATA &X01000000      ' █
1741 DATA &X01011100      ' █ ███
1742 DATA &X01100010      ' ██   █
1743 DATA &X01000010      ' █    █
1744 DATA &X01100010      ' ██   █
1745 DATA &X01011100      ' █ ███
1746 DATA &X00000000      '
1747 '
1748 'c
1749 '
1750 DATA &X00000000      '
1751 DATA &X00000000      '
```

```
1752 DATA &X00111100        '  ▮▮▮▮
1753 DATA &X01000010        ' ▮    ▮
1754 DATA &X01000000        ' ▮
1755 DATA &X01000010        ' ▮    ▮
1756 DATA &X00111100        '  ▮▮▮▮
1757 DATA &X00000000        '
1758 '
1759 'd
1760 '
1761 DATA &X00000010        '      ▮
1762 DATA &X00000010        '      ▮
1763 DATA &X00111010        '  ▮▮▮ ▮
1764 DATA &X01000110        ' ▮   ▮▮
1765 DATA &X01000010        ' ▮    ▮
1766 DATA &X01000110        ' ▮   ▮▮
1767 DATA &X00111010        '  ▮▮▮ ▮
1768 DATA &X00000000        '
1769 '
1770 'e
1771 '
1772 DATA &X00000000        '
1773 DATA &X00000000        '
1774 DATA &X00111100        '  ▮▮▮▮
1775 DATA &X01000010        ' ▮    ▮
1776 DATA &X01111110        ' ▮▮▮▮▮▮
1777 DATA &X01000000        ' ▮
1778 DATA &X00111100        '  ▮▮▮▮
1779 DATA &X00000000        '
1780 '
1781 'f
1782 '
1783 DATA &X00001100        '    ▮▮
1784 DATA &X00010010        '   ▮  ▮
1785 DATA &X00010000        '   ▮
1786 DATA &X01111100        ' ▮▮▮▮▮
1787 DATA &X00010000        '   ▮
1788 DATA &X00010000        '   ▮
1789 DATA &X00010000        '   ▮
1790 DATA &X00000000        '
1791 '
1792 'g
1793 '
```

```
1794 DATA &X00000000      '
1795 DATA &X00000000      '
1796 DATA &X00111010      '  ▮▮▮ ▮
1797 DATA &X01000110      ' ▮   ▮▮
1798 DATA &X01000110      ' ▮   ▮▮
1799 DATA &X00111010      '  ▮▮▮ ▮
1800 DATA &X00000010      '      ▮
1801 DATA &X00111100      '  ▮▮▮▮
1802 '
1803 'h
1804 '
1805 DATA &X01000000      ' ▮
1806 DATA &X01000000      ' ▮
1807 DATA &X01011100      ' ▮ ▮▮▮
1808 DATA &X01100010      ' ▮▮   ▮
1809 DATA &X01000010      ' ▮    ▮
1810 DATA &X01000010      ' ▮    ▮
1811 DATA &X01000010      ' ▮    ▮
1812 DATA &X00000000      '
1813 '
1814 'i
1815 '
1816 DATA &X00001000      '    ▮
1817 DATA &X00000000      '
1818 DATA &X00011000      '   ▮▮
1819 DATA &X00001000      '    ▮
1820 DATA &X00001000      '    ▮
1821 DATA &X00001000      '    ▮
1822 DATA &X00011100      '   ▮▮▮
1823 DATA &X00000000      '
1824 '
1825 'j
1826 '
1827 DATA &X00000100      '     ▮
1828 DATA &X00000000      '
1829 DATA &X00001100      '    ▮▮
1830 DATA &X00000100      '     ▮
1831 DATA &X00000100      '     ▮
1832 DATA &X00000100      '     ▮
1833 DATA &X01000100      ' ▮   ▮
1834 DATA &X00111000      '  ▮▮▮
1835 '
```

```
1836 'k
1837 '
1838 DATA &X01000000        ' ■
1839 DATA &X01000000        ' ■
1840 DATA &X01000100        ' ■   ■
1841 DATA &X01001000        ' ■  ■
1842 DATA &X01010000        ' ■ ■
1843 DATA &X01101000        ' ■■ ■
1844 DATA &X01000100        ' ■   ■
1845 DATA &X00000000        '
1846 '
1847 'l
1848 '
1849 DATA &X00011000        '   ■■
1850 DATA &X00001000        '    ■
1851 DATA &X00001000        '    ■
1852 DATA &X00001000        '    ■
1853 DATA &X00001000        '    ■
1854 DATA &X00001000        '    ■
1855 DATA &X00011100        '   ■■■
1856 DATA &X00000000        '
1857 '
1858 'm
1859 '
1860 DATA &X00000000        '
1861 DATA &X00000000        '
1862 DATA &X01110110        ' ■■■ ■■
1863 DATA &X01001001        ' ■  ■  ■
1864 DATA &X01001001        ' ■  ■  ■
1865 DATA &X01001001        ' ■  ■  ■
1866 DATA &X01001001        ' ■  ■  ■
1867 DATA &X00000000        '
1868 '
1869 'n
1870 '
1871 DATA &X00000000        '
1872 DATA &X00000000        '
1873 DATA &X01011100        ' ■ ■■■
1874 DATA &X01100010        ' ■■   ■
1875 DATA &X01000010        ' ■    ■
1876 DATA &X01000010        ' ■    ■
1877 DATA &X01000010        ' ■    ■
```

```
1878 DATA &X00000000      '
1879 '
1880 'o
1881 '
1882 DATA &X00000000      '
1883 DATA &X00000000      '
1884 DATA &X00111100      '  ████
1885 DATA &X01000010      ' █    █
1886 DATA &X01000010      ' █    █
1887 DATA &X01000010      ' █    █
1888 DATA &X00111100      '  ████
1889 DATA &X00000000      '
1890 '
1891 'p
1892 '
1893 DATA &X00000000      '
1894 DATA &X00000000      '
1895 DATA &X01011100      ' █ ███
1896 DATA &X01100010      ' ██   █
1897 DATA &X01100010      ' ██   █
1898 DATA &X01011100      ' █ ███
1899 DATA &X01000000      ' █
1900 DATA &X01000000      ' █
1901 '
1902 'q
1903 '
1904 DATA &X00000000      '
1905 DATA &X00000000      '
1906 DATA &X00111010      '  ███ █
1907 DATA &X01000110      ' █   ██
1908 DATA &X01000110      ' █   ██
1909 DATA &X00111010      '  ███ █
1910 DATA &X00000010      '      █
1911 DATA &X00000010      '      █
1912 '
1913 'r
1914 '
1915 DATA &X00000000      '
1916 DATA &X00000000      '
1917 DATA &X01011100      ' █ ███
1918 DATA &X01100010      ' ██   █
1919 DATA &X01000000      ' █
```

```
1920 DATA &X01000000       ' ▮
1921 DATA &X01000000       ' ▮
1922 DATA &X00000000       '
1923 '
1924 's
1925 '
1926 DATA &X00000000       '
1927 DATA &X00000000       '
1928 DATA &X00111110       '  ▮▮▮▮▮
1929 DATA &X01000000       ' ▮
1930 DATA &X00111100       '  ▮▮▮▮
1931 DATA &X00000010       '      ▮
1932 DATA &X01111100       ' ▮▮▮▮▮
1933 DATA &X00000000       '
1934 '
1935 't
1936 '
1937 DATA &X00010000       '   ▮
1938 DATA &X00010000       '   ▮
1939 DATA &X01111100       ' ▮▮▮▮▮
1940 DATA &X00010000       '   ▮
1941 DATA &X00010000       '   ▮
1942 DATA &X00010010       '   ▮  ▮
1943 DATA &X00001100       '    ▮▮
1944 DATA &X00000000       '
1945 '
1946 'u
1947 '
1948 DATA &X00000000       '
1949 DATA &X00000000       '
1950 DATA &X01000010       ' ▮    ▮
1951 DATA &X01000010       ' ▮    ▮
1952 DATA &X01000010       ' ▮    ▮
1953 DATA &X01000110       ' ▮   ▮▮
1954 DATA &X00111010       '  ▮▮▮ ▮
1955 DATA &X00000000       '
1956 '
1957 'v
1958 '
1959 DATA &X00000000       '
1960 DATA &X00000000       '
1961 DATA &X01000010       ' ▮    ▮
```

```
1962 DATA &X01000010       ' ▮    ▮
1963 DATA &X01000010       ' ▮    ▮
1964 DATA &X00100100       '  ▮  ▮
1965 DATA &X00011000       '   ▮▮
1966 DATA &X00000000       '
1967 '
1968 'w
1969 '
1970 DATA &X00000000       '
1971 DATA &X00000000       '
1972 DATA &X01000001       ' ▮     ▮
1973 DATA &X01001001       ' ▮  ▮  ▮
1974 DATA &X01001001       ' ▮  ▮  ▮
1975 DATA &X01001001       ' ▮  ▮  ▮
1976 DATA &X00110110       '  ▮▮ ▮▮
1977 DATA &X00000000       '
1978 '
1979 'x
1980 '
1981 DATA &X00000000       '
1982 DATA &X00000000       '
1983 DATA &X01000010       ' ▮    ▮
1984 DATA &X00100100       '  ▮  ▮
1985 DATA &X00011000       '   ▮▮
1986 DATA &X00100100       '  ▮  ▮
1987 DATA &X01000010       ' ▮    ▮
1988 DATA &X00000000       '
1989 '
1990 'y
1991 '
1992 DATA &X00000000       '
1993 DATA &X00000000       '
1994 DATA &X01000010       ' ▮    ▮
1995 DATA &X01000010       ' ▮    ▮
1996 DATA &X01000110       ' ▮   ▮▮
1997 DATA &X00111010       '  ▮▮▮ ▮
1998 DATA &X00000010       '      ▮
1999 DATA &X00111100       '  ▮▮▮▮
2000 '
2001 'z
2002 '
2003 DATA &X00000000       '
```

```
2004 DATA &X00000000      '
2005 DATA &X01111110      ' ██████
2006 DATA &X00000100      '      █
2007 DATA &X00011000      '    ██
2008 DATA &X00100000      '   █
2009 DATA &X01111110      ' ██████
2010 DATA &X00000000      '
2011 '
2012 '{
2013 '
2014 DATA &X00001100      '     ██
2015 DATA &X00010000      '    █
2016 DATA &X00001000      '     █
2017 DATA &X00110000      '   ██
2018 DATA &X00001000      '     █
2019 DATA &X00010000      '    █
2020 DATA &X00001100      '     ██
2021 DATA &X00000000      '
2022 '
2023 '¦
2024 '
2025 DATA &X00001000      '     █
2026 DATA &X00001000      '     █
2027 DATA &X00001000      '     █
2028 DATA &X00001000      '     █
2029 DATA &X00001000      '     █
2030 DATA &X00001000      '     █
2031 DATA &X00001000      '     █
2032 DATA &X00000000      '
2033 '
2034 '}
2035 '
2036 DATA &X00110000      '   ██
2037 DATA &X00001000      '     █
2038 DATA &X00010000      '    █
2039 DATA &X00001100      '     ██
2040 DATA &X00010000      '    █
2041 DATA &X00001000      '     █
2042 DATA &X00110000      '   ██
2043 DATA &X00000000      '
```

Description du programme :

Après l'en-tête du programme, la ligne 1007 autorise avec l'instruction SYMBOL AFTER 33 la redéfinition des caractères à partir du code 33. La boucle FOR-TO-NEXT des lignes 1009 à 1018 modifie le jeu de caractères. La boucle extérieure produit les codes des caractères qui doivent être redéfinis. Une seconde boucle FOR-TO-NEXT lit ensuite dans les instructions DATA huit valeurs numériques contenant la matrice de caractère pour les affecter aux variables indicées TX(0) à TX(7). La matrice est enfin intégrée dans le jeu de caractères (ligne 1016) sous le code de caractère actuel (I). On procède de même avec tous les caractères. Une fois la redéfinition terminée, le programme met à votre disposition le nouveau jeu de caractères et se supprime lui-même de la mémoire (ligne 1020)!

De la ligne 1021 à la fin du programme figurent des instructions DATA par groupes de huit contenant chacun la matrice d'un caractère. Pour introduire un minimum de structure dans cette masse de DATAs, chaque groupe est séparé du suivant par des lignes de commentaires qui indiquent chaque fois quel caractère est codifié par la matrice suivante. Pour obtenir un résultat plus parlant, nous avons opté pour une représentation binaire des matrices de caractère. Les commentaires dans les lignes de DATA servent uniquement à faciliter l'examen par le lecteur et ils ne doivent pas être tapés dans le programme.

3.7 DESTROYED - UN JEU D'ARCADE EN BASIC

Les jeux électroniques du style "Space Invaders", "Defender" ou "Pac-Man" ont fait époque au milieu des années soixante-dix dans le domaine de l'industrie du divertissement. C'était en effet la première fois qu'on arrivait à utiliser l'ordinateur à des fins commerciales. Au fur et à mesure que la passion de ces jeux s'emparait de certaines couches de la population, apparut un véritable "boom" des jeux électroniques. Ce boom joua d'ailleurs un rôle décisif dans la progression fulgurante du marché des ordinateurs familiaux.

On peut trouver aujourd'hui des jeux électroniques pour pratiquement n'importe quel système informatique.

Le choix de jeux électroniques disponibles sur le CPC est également et notamment tout à fait considérable. Ce plaisir ludique n'est malheureusement pas précisément bon marché et de nombreux possesseurs d'ordinateurs renoncent pour cette raison à utiliser leur machine pour jouer. Essayons de remédier à cette situation!

Bien que le CPC AMSTRAD n'ait pas été conçu spécialement pour les jeux électroniques, il permet de développer des jeux électroniques personnels aussi bien en BASIC qu'en langage machine grâce à ces remarquables possibilités graphiques. Il est d'ailleurs tout à fait suprenant de constater à quel point même un jeu réalisé en BASIC permet un déroulement rapide et intéressant, autorisant de ce fait une grande diversité d'actions, grâce au jeu d'instructions BASIC très complet dont disposent tous les CPCs. C'est d'ailleurs justement la richesse de ce jeu d'instructions qui permet même au débutant de réaliser sans trop de travail des jeux électroniques d'un bon niveau. Nous allons maintenant essayer de vous expliquer comment et grâce à quelles techniques on peut convertir une idée de jeu en un programme. A cet égard, la définition de caractères que nous venons de vous expliquer jouera un rôle très important. Tous les objets que nous rencontrerons dans notre jeu seront en effet produits à partir de caractères redéfinis. Les gros objets seront simplement composés de différents éléments (un peu comme une mosaïque) dont chacun sera à son tour un caractère redéfini.

3.7.1 L'idée de jeu

Chaque jeu électronique est basé sur une idée de jeu. Il est inutile de commencer la création d'un jeu électronique en tapant des lignes de BASIC "en veux-tu, en voilà". Les tentatives de ce genre débouchent généralement sur un désordre complet qui les condamne à un échec définitif. Tout programmeur de jeu a besoin d'une idée de base et d'un but de jeu, d'un fil conducteur qu'il pourra réaliser point par point dans son programme. Il est facile de trouver des idées de ce type. Vous pouvez utiliser par exemple comme modèle de votre jeu électronique un film, l'histoire d'un livre riche en rebondissements ou un jeu électronique commercial. N'oubliez pas toutefois dans ce cas que vous n'avez le droit de commercialiser un jeu que si l'idée qui est à la base de ce jeu a été imaginée par vous-même. Vous pouvez donc donner libre cours à votre imagination.

Le jeu électronique que nous souhaitons vous présenter s'appelle "DESTROYED" et il appartient à la catégorie des jeux de l'espace. La trame peut être retracée rapidement.

> "Quelque part dans l'univers se trouve une station spatiale gigantesque. Des pirates ennemis veulent la détruire pour prendre le contrôle du système solaire. Il s'agit donc maintenant de repousser les pirates à l'aide des quatre canons de bord géants dont vous disposez. Comme les batteries du système de défense de la station spatiale ne sont pas conçues pour supporter l'emploi simultané de tous les canons laser, on ne peut tirer qu'avec une seule arme à la fois. L'action globale est en outre rendue plus ardue du fait de la provision d'énergie limitée dont vous disposez.
>
> Les pirates attaquent par quatre vagues offensives venant de tous les côtés. Ils essaient tout d'abord, dans la plus pure tradition kamikaze, de percuter la station avec leurs vaisseaux. Ils emploient ensuite des torpilles de photons. Dans les deux dernières vagues offensives, les pirates combinent leurs stratégies et attaquent avec tout ce dont ils disposent. Le tout se termine dans une gigantesque confrontation de projectiles dans laquelle le commandant de notre station spatiale, c'est-à-dire le joueur, doit tenter de garder la tête froide pour ne pas gaspiller l'énergie disponible en tirant dans tous les sens. Chaque coup doit porter! Si cette bataille a elle aussi pu être gagnée, le héros victorieux est soumis à un entraînement inoffensif (un tour de bonus) qui peut lui permettre d'améliorer son score. Toute cette magie recommence ensuite avec un niveau de difficulté accru..."

Pour repousser les vaisseaux pirates ennemis, le joueur doit activer le canon laser voulu avec la touche fléchée correspondant à sa direction. Vous tirez avec la barre espace!

3.7.2 Construction des scénarios

Comme vous pouvez le voir d'après l'idée de jeu, un tour de jeu se compose de cinq scénarios; quatre vagues d'agression et un tour bonus. Il faut maintenant réaliser graphiquement chacune de ces

scènes du jeu. Pour que nous puissions nous faire une idée des objets nécessaires pour le déroulement du jeu, l'action de chaque scénario doit être définie. Les différentes scènes de jeu se définissent l'une par rapport à l'autre de la façon suivante :

Scène 1 (Destruction Wave) :

Des vaisseaux ennemis attaquent et il faut les détruire.

Scène 2 (Missile Wave) :

Il faut détruire des projectiles ennemis.

Scène 3 (Survival Wave) :

Il faut détruire des vaisseaux et projectiles ennemis. Si la station du joueur n'est pas détruite ici, il reçoit un bonus.

Scène 4 (Final Wave) :

Il faut détruire des vaisseaux et projectiles ennemis. Le joueur reçoit à la fin de la vague offensive un bonus en fonction de l'énergie restante.

Scène 5 (Bonus Round) :

Il faut éliminer autant de projectiles ennemis que possible. La station spatiale ne peut être détruite pendant ce tour.

Cet aperçu nous permet maintenant d'établir une liste de tous les objets nécessaires :

1. une station spatiale
2. quatre vaisseaux pirates (un vaisseau sous quatre angles différents)
3. une torpille spatiale (projectile)

Il faut en outre que les stations restantes soient chacune identifiées par un drapeau et qu'elles reçoivent une forme futuriste à l'aide des caractères ASCII 48 à 122.

Nous conseillons à l'utilisateur d'effectuer la redéfinition de ce grand nombre de caractères au moyen de l'éditeur de caractères présenté au chapitre 3.4. Il est conçu de telle façon qu'il écrit automatiquement sur la disquette le jeu de caractères redéfini sous forme de lignes SYMBOL dans un programme BASIC généré automatiquement. Ces données peuvent ensuite, dans une autre étape du travail, être intégrées sans difficulté dans le programme principal à l'aide de l'instruction MERGE.

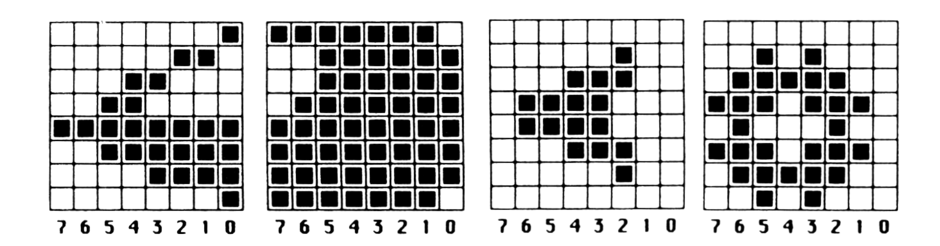


Figure 6 : **Vaisseau spatial ennemi et mine**

Après la définition de tous les objets apparaissant dans le jeu, il s'agit maintenant de créer un ou plusieurs dessins d'arrière-plans dans l'esprit du jeu. Pour la réalisation d'images d'arrière-plans complexes, nous conseillons d'utiliser un éditeur graphique car cela permet de réduire considérablement le temps nécessaire à la réalisation d'une telle image. On peut trouver de tels éditeurs (programmes de dessin) graphiques dans le commerce, souvent même pour moins de 300 F. Une autre possibilité consiste naturellement à développer vous-même l'éditeur graphique. Vous pouvez vous baser pour cela sur le programme de dessin décrit à la section 2.5. Une fois que tous les dessins d'arrière-plans nécessaires ont été réalisés ils peuvent être à tout moment chargés dans la mémoire écran par le programme de jeu avec l'instruction LOAD.

Des images simples peuvent aussi être créées directement à partir du programme de jeu lui-même.

Notre jeu de bataille spatiale marque par exemple sur l'écran à l'aide de la fonction aléatoire RND, avec la ligne

```
1500 FOR i=1 TO 100:PLOT(RND(1)*639),(RND(1)*399),3:NEXT
```

cent points blancs qui symboliseront les étoiles. On porte ensuite sur l'écran les inscriptions suivantes

UP
BEST (record) et
LEVEL (vague d'agression)

La ligne

```
1770 FOR EN=221 TO 421 STEP 4:MOVE EN,400:DRAW EN,384,2:NEXT
```

place ensuite dans la partie supérieure de l'écran un indicateur de niveau énergétique composé de plusieurs traits consécutifs.

3.7.3 Animation

Il s'agit maintenant d'animer sur l'écran, par programme, les caractères définis et intégrés dans le programme de jeu. La routine que nous utilisons résout ce problème en suivant une méthode simple : tous les objets sont déplacés dans une boucle principale dont la variable de comptage contient la valeur de position à modifier. Ils sont pour cela placés sur l'écran suivis d'un espace. Lorsque l'objet est déplacé en avant d'un caractère lors du prochain parcours de boucle, l'objet ou le dernier caractère de l'objet sera automatiquement effacé par l'espace suivant la chaîne de caractères.

Les rayons laser avec lesquels le joueur peut tirer sur les objets ennemis ne sont rien d'autre que quatre instructions DRAW, une seule d'entre elles étant appelée à la fois. Comme la ligne est dessinée avec un PEN auquel sont attribuées deux paramètres INK, on obtient une impression de rayon fluorescent.

Pour que vous ne vous découragiez pas en tapant ce listing de programme vraiment long, nous avons illustré trois phases du programme par des copies d'écran.



DESTROYED IS WRITTEN 1986 BY JST & TAV

Copie d'écran 4 : Image-titre

DESTROYED

YOUR MISSION IS TO DESTROY ALIENS,
GOOD LUCK!

MINE: 50 PTS
ALIEN SHIP: 100 PTS
EXTRA BASE: 20000 PTS

PRESS ANY KEY TO INSERT COIN

Copie d'écran 5 : Tableau des scores

1 UP 000400 BEST 000000

LEVEL 1

Copie d'écran 7 : Situation de Jeu

```
100 MODE 1
110 BORDER 0:SPEED KEY 255,255
120 DEFINT A-Z:DIM KN(3,5),HI!(9),INI$(9)
130 FOR I=0 TO 9:HI!(I)=1000:INI$(I)="XXX":NEXT
140 RANDOMIZE TIME
150 WV$(0)="DESTRUCTION WAVE":WV$(1)="MISSILE
WAVE":WV$(2)="SURVIVALWAVE":WV$(3)="FINALWAVE":WV$(4)="BONUS
ROUND"
160 '
170 'DATAs DANS LES TABLEAUX
180 '
190 FOR I=0 TO 3:FOR J=0 TO 5:READ X:KN(I,J)=X:NEXT:NEXT
200 '
```

```
210 SYMBOL AFTER 48
220 SYMBOL 48,254,206,206,214,230,230,254,0
230 SYMBOL 49,56,24,24,24,126,126,126,0
240 SYMBOL 50,254,6,6,254,192,192,254,0
250 SYMBOL 51,124,12,12,126,14,14,254,0
260 SYMBOL 52,224,224,224,236,238,254,14,0
270 SYMBOL 53,254,192,192,254,14,14,254,0
280 SYMBOL 54,248,216,192,254,206,206,254,0
290 SYMBOL 55,254,6,6,62,56,56,56,0
300 SYMBOL 56,252,204,204,254,206,206,254,0
310 SYMBOL 57,254,198,198,254,14,14,14,0
320 SYMBOL 65,126,102,102,254,230,230,230,0
330 SYMBOL 66,252,204,204,254,206,206,254,0
340 SYMBOL 67,254,206,206,192,198,198,254,0
350 SYMBOL 68,252,204,204,238,238,238,254,0
360 SYMBOL 69,254,192,192,254,224,224,254,0
370 SYMBOL 70,254,192,192,254,224,224,224,0
380 SYMBOL 71,254,198,192,222,206,206,254,0
390 SYMBOL 72,198,198,198,254,206,206,206,0
400 SYMBOL 73,24,24,24,28,28,28,28,0
410 SYMBOL 74,12,12,12,14,14,110,126,0
420 SYMBOL 75,204,216,216,254,206,206,206,0
430 SYMBOL 76,48,48,48,112,112,112,126,0
440 SYMBOL 77,206,254,254,238,206,206,206,0
450 SYMBOL 78,206,238,254,222,206,206,206,0
460 SYMBOL 79,254,198,198,206,206,206,254,0
470 SYMBOL 80,254,198,198,254,224,224,224,0
480 SYMBOL 81,252,204,204,204,206,222,254,0
490 SYMBOL 82,252,198,198,252,206,206,206,0
500 SYMBOL 83,254,192,254,6,230,230,254,0
510 SYMBOL 84,254,48,48,48,56,56,56,0
520 SYMBOL 85,206,206,206,206,206,206,254,0
530 SYMBOL 86,206,206,206,206,222,124,56,0
540 SYMBOL 88,230,230,230,124,206,206,206,0
550 SYMBOL 89,206,206,206,254,56,56,56,0
560 SYMBOL 90,254,204,216,48,110,206,254,0
570 '
580 SYMBOL 240,0,0,0,0,40,108,238,238
590 SYMBOL 241,238,238,108,40,0,0,0,0
600 SYMBOL 242,0,3,7,15,0,15,7,3
610 SYMBOL 243,0,192,224,240,0,240,224,192
620 SYMBOL 244,16,16,16,16,56,124,254,254
```

```
630 SYMBOL 245,254,254,124,56,16,16,16,16
640 SYMBOL 246,0,3,7,15,255,15,7,3
650 SYMBOL 247,0,192,224,240,255,240,224,192
660 '
670 SYMBOL 207,1,6,24,48,255,63,15,1
680 SYMBOL 208,254,63,63,127,255,255,255,254
690 SYMBOL 209,0,4,28,120,120,28,4,0
700 '
710 SYMBOL 204,143,78,78,46,60,28,8,8
720 SYMBOL 205,126,255,255,255,255,255,159,143
730 SYMBOL 206,0,0,102,60,60,24,24,0
740 '
750 SYMBOL 201,8,8,28,60,46,78,78,143
760 SYMBOL 202,143,159,255,255,255,255,255,126
770 SYMBOL 203,0,24,24,60,60,102,0,0
780 '
790 SYMBOL 210,128,96,24,12,255,252,240,128
800 SYMBOL 211,127,252,252,254,255,255,255,127
810 SYMBOL 212,0,32,56,30,30,56,32,0
820 '
830 SYMBOL 213,0,40,86,84,168,54,212,42
840 SYMBOL 214,0,40,124,238,68,238,124,40
850 '
860 SYMBOL 215,3,7,7,7,7,127,255,255
870 SYMBOL 216,255,207,215,219,219,219,219,219
880 SYMBOL 217,239,239,239,239,239,239,239,239
890 SYMBOL 218,255,243,235,219,219,219,219,219
900 SYMBOL 219,192,224,224,224,224,254,255,255
910 SYMBOL 220,255,255,128,191,223,224,255,255
920 SYMBOL 221,219,219,27,235,243,3,255,255
930 SYMBOL 222,239,239,199,199,199,199,199,1
940 SYMBOL 223,219,219,216,215,207,192,255,255
950 SYMBOL 224,255,255,1,253,251,7,255,255
960 SYMBOL 225,255,255,255,255,0,255,255,255
970 SYMBOL 226,254,252,252,129,0,129,252,252
980 SYMBOL 227,40,238,198,131,16,131,198,238
990 SYMBOL 228,255,127,127,3,0,3,127,127
1000 SYMBOL 229,255,255,255,255,0,255,255,255
1010 SYMBOL 230,255,255,255,224,223,191,128,255
1020 SYMBOL 231,254,255,255,3,243,235,27,219
1030 SYMBOL 232,40,1,199,199,199,199,199,239
1040 SYMBOL 233,255,255,255,192,207,215,216,219
```

```
1050 SYMBOL 200,96,120,126,126,78,64,64,64
1060 SYMBOL 234,255,255,255,7,251,253,1,255
1070 SYMBOL 235,255,255,127,7,7,7,7,3
1080 SYMBOL 236,219,219,219,219,219,215,207,255
1090 SYMBOL 237,239,239,239,239,239,239,239,239
1100 SYMBOL 238,219,219,219,219,219,235,243,255
1110 SYMBOL 239,255,255,254,224,224,224,224,192
1120 '
1130 'INTRO
1140 '
1150 LBX=320: LBY=16
1160 SPEED INK 4,4:INK 0,0:INK 1,24:INK 2,11,6:INK 3,13:PEN 3:CLS
1170 LOCATE 1,25: PRINT" DESTROYED IS WRITTEN 1986 BY JST & TAV"
1180 TAG
1190 FOR Y=7 TO 1 STEP -1
1200 FOR X=0 TO 72
1210 IF TEST(X*2+16,Y*2)=3 THEN LPX=X*8+32: LPY=Y*8+230:MOVE
LBX,LBY:DRAW LPX+4,LPY-4,2:MOVE LBX,LBY:DRAW LPX+4,LPY-4,0:PLOT
LPX,LPY,1:PRINT CHR$(129);
1220 IF INKEY$<>"" THEN 1430
1230 NEXT X
1240 NEXT Y
1250 INK 1,6,11:FOR I=1 TO 3000:NEXT
1260 TAGOFF
1270 LOCATE 1,25:PRINT SPACE$(40)
1280 FOR I=1 TO 5:PRINT:NEXT
1290 INK 3,24:PEN 3
1300 LOCATE 4,10:PRINT"YOUR MISSION IS TO DESTROY ALIENS,":LOCATE
23,12:PRINT"GOOD LUCK!"
1310 LOCATE 7,17:PRINT"MINE:"
1320 LOCATE 7,19:PRINT"ALIEN SHIP:"
1330 LOCATE 7,21:PRINT"EXTRA BASE:"
1340 INK 2,13:PEN 2:FOR I=40 TO 20 STEP-1:LOCATE I,17:PRINT
CHR$(214);" ";:FOR J=1 TO 300:NEXT:NEXT:PEN 3:PRINT"    50 PTS"
1350 FOR I=38 TO 19 STEP-1:LOCATE I,19:PEN 3:PRINT
CHR$(207);CHR$(208);:PEN 1:PRINT CHR$(209);" ";:FOR J=1 TO
300:NEXT:NEXT:PEN 3:PRINT"  100 PTS"
1360 INK 2,13:PEN 2:FOR I=40 TO 20 STEP-1:LOCATE I,21:PRINT
CHR$(200);" ";:FOR J=1 TO 300:NEXT:NEXT:PEN 3:PRINT" 20000 PTS"
1370 PEN 2:LOCATE 6,25:PRINT"PRESS ANY KEY TO INSERT COIN";
1380 FOR I=1 TO 15000:IF INKEY$="" THEN NEXT ELSE 1430
1390 CLS:INK 2,6:LOCATE 13,1:PRINT"ALL TIME HEROS:":INK 3,1
```

```
1400 INK 1,18,24:SPEED INK 15,15:FOR I=0 TO 9:LOCATE 11,5+I*2:PRINT
USING"##";I+1;:PRINT".  ";:PEN 1:PRINT USING"######";HI!(I);:PEN
3:PRINT SPC(3);INI$(I):PEN 2:NEXT
1410 FOR I=1 TO 15000:IF INKEY$="" THEN NEXT ELSE 1430
1420 CLS:GOTO 1160
1430 '
1440 TAGOFF:CLS:SPEED INK 4,4:INK 0,0:INK 1,24:INK 2,11,6:INK 3,13
1450 '
1460 LEVEL=0:OFS=-5:WAVE=-1:PP=1:BASE=3::FB=1:AS=0
1470 '
1480 'LE CIEL ET LES ETOILES
1490 '
1500 FOR i=1 TO 100:PLOT(RND(1)*639),(RND(1)*399),3:NEXT
1510 '
1520 'LE CENTRE ENERGETIQUE
1530 '
1540 PEN 2
1550 LOCATE 20,11:PRINT CHR$(143)
1560 LOCATE 20,12:PRINT CHR$(143)
1570 LOCATE 18,13:PRINT STRING$(5,143)
1580 LOCATE 20,14:PRINT CHR$(143)
1590 LOCATE 20,15:PRINT CHR$(143)
1600 '
1610 'CONSTRUCTION DE LA STATION
1620 '
1630 PEN 1
1640 PRINT CHR$(22);CHR$(1)
1650 S0=0:FOR i=1 TO 5:LOCATE 18,i+10:FOR j=215+S0 TO 219+S0:PRINT
CHR$(j);:NEXT:S0=S0+5:NEXT
1660 PRINT CHR$(22);CHR$(0)
1670 LOCATE 17,13:PRINT CHR$(242);
1680 LOCATE 23,13:PRINT CHR$(243);
1690 LOCATE 20,10:PRINT CHR$(244);
1700 LOCATE 20,16:PRINT CHR$(241);
1710 '
1720 PEN 3:LOCATE 1,1:PRINT"1 UP 000000";SPC(18);"BEST";:LOCATE
35,1:SCORE!=BEST!:GOSUB 2570:SCORE!=0:LOCATE 1,25:PRINT"LEVEL
1":LOCATE 39,25:PRINT STRING$(2,200);:PEN 1
1730 '
1740 'BOUCLE PRINCIPALE
1750 '
1760 LEVEL=LEVEL+1:PEN 3:LOCATE 6,25:PRINT LEVEL
```

```
1770 FOR EN=221 TO 421 STEP 4:MOVE EN,400:DRAW EN,384,2:NEXT
1780 OFS=OFS+5:IF OFS>20 THEN OFS=20
1790 '
1800 SHPC=30+OFS:WAVE=WAVE+1:IF WAVE>4 THEN WAVE=-1:GOTO 1760
1810  IF WAVE=4 THEN I=EN:FOR I=EN TO 220 STEP-1:MOVE I,400:DRAW
I,384,0:NEXT:EN=(EN-220)*10:TEXT$="FUEL BONUS ="+STR$(EN):GOSUB
2540:SCORE!=SCORE!+EN:GOSUB 2550:INK 1,0 ELSE INK 1,24
1820 IF WAVE=3 THEN IF SUR=1 THEN TEXT$="SURVIVAL POINTS
="+STR$(SHPC*50):GOSUB 2540:SCORE!=SCORE!+SHPC*50:GOSUB 2550 ELSE
TEXT$="SORRY, NO BONUS":GOSUB 2540
1830 IF WAVE=2 THEN INK 0,1:BORDER 1 ELSE INK 0,0:BORDER 0:SUR=1
1840 TEXT$=WV$(WAVE):GOSUB 2540
1850 '
1860 GOSUB 2280
1870 IF OX<17 OR OX>23 OR OY<10 OR OY>16 THEN 1930
1880 IF WAVE=4 THEN AS=KNX+1:PP=0:GOTO 2100
1890 BASE=BASE-1:SUR=SUR-1:IF BASE<>0 THEN GOSUB 2580
1900 INK 0,27,0:FOR i=1 TO 1000:NEXT:IF WAVE=2 THEN INK 0,1 ELSE
INK 0,0
1910 GOSUB 2550:IF BASE=0 THEN 2590 ELSE TEXT$="GET READY!":GOSUB
2540:KNX=AS-1:IF KNX>3 THEN KNX=KNX-4
1920 GOTO 2100
1930 Y$=INKEY$
1940 IF Y$=CHR$(32) THEN 2030
1950 IF y$<CHR$(240) OR Y$>CHR$(243) THEN 1860
1960 '
1970 'FIXER CANON
1980 '
1990 LOCATE KN(KNX,0),KN(KNX,1):PRINT CHR$(KNX+240);
2000 KNX=ASC(Y$)-240
2010 LOCATE KN(KNX,0),KN(KNX,1):PRINT CHR$(KNX+244);
2020 GOTO 1860
2030 '
2040 'TIR
2050 '
2060 SOUND 1,4000,40,11,5,3,7
2070 IF WAVE=4 THEN 2090
2080 EN=EN-1:MOVE EN,400:DRAW EN,384,0:IF EN-220<0 THEN 2590
2090 MOVE KN(KNX,2),KN(KNX,3):DRAW KN(KNX,4),KN(KNX,5),2:FOR I=1 TO
20:NEXT:DRAW KN(KNX,2),KN(KNX,3),0
2100 IF KNX+1=AS THEN LOCATE OX,OY:PEN 2:PRINT CHR$(213);:GOSUB
2190:LOCATE OX,OY:PRINT" ";:IF PP THEN SCORE!=SCORE!+50:GOTO 2170
```

```
ELSE 2180
2110 IF KNX+5<>AS THEN 1860 ELSE LOCATE OX,OY:PEN 2
2120 IF AS=5 THEN PRINT
CHR$(213);CHR$(11);CHR$(8);CHR$(213);CHR$(11);CHR$(8);:GOSUB
2190:PRINT " ";CHR$(10);CHR$(8);" ";CHR$(10);CHR$(8);" ";
2130 IF AS=6 THEN PRINT
CHR$(213);CHR$(10);CHR$(8);CHR$(213);CHR$(10);CHR$(8);:GOSUB
2190:PRINT " ";CHR$(11);CHR$(8);" ";CHR$(11);CHR$(8);" ";
2140 IF AS=7 THEN PRINT CHR$(8);CHR$(213);CHR$(213);:GOSUB
2190:PRINT STRING$(3,8);STRING$(3,32);
2150 IF AS=8 THEN PRINT CHR$(213);CHR$(213);:GOSUB 2190:PRINT
STRING$(2,8);STRING$(3,32);
2160 SCORE!=SCORE!+100 ELSE 2180
2170 GOSUB 2550
2180 PEN 1:AS=0:PP=1:SHPC=SHPC-1:IF SHPC=0 THEN 1800 ELSE 1860
2190 FOR I=1 TO 100:NEXT:RETURN
2200 '
2210 'DATAs AVEC INDICATIONS DE POSITION
2220 '
2230 DATA 20,10,310,255,310,399
2240 DATA 20,16,310,143,310,0
2250 DATA 17,13,256,198,0,198
2260 DATA 23,13,368,198,639,198
2270 '
2280 IF AS<>0 THEN 2410
2290 AS=INT(RND(1)*4)+1
2300 IF WAVE=1 OR WAVE=4 THEN 2330
2310 IF WAVE=0 THEN AS=AS+4:GOTO 2330
2320 AS=INT(RND(1)*8)+1
2330 IF AS=3 THEN AX=1:AY=13
2340 IF AS=4 THEN AX=40:AY=13
2350 IF AS=1 THEN AX=20:AY=2
2360 IF AS=2 THEN AX=20:AY=25
2370 IF AS=7 THEN AX=1:AY=13
2380 IF AS=8 THEN AX=38:AY=13
2390 IF AS=5 THEN AX=20:AY=3
2400 IF AS=6 THEN AX=20:AY=22
2410 LOCATE AX,AY
2420 IF AS=3 THEN:PEN 3:PRINT" ";CHR$(214);:AX=AX+1:OX=AX:OY=AY
2430 IF AS=4 THEN:PEN 3:OX=AX:OY=AY:PRINT CHR$(214);" ";:AX=AX-1
2440 IF AS=1 THEN PEN 3:PRINT" ";:AY=AY+1:LOCATE AX,AY:PRINT
CHR$(214);:OX=AX:OY=AY
```

```
2450 IF AS=2 THEN PEN 3:PRINT" ";:AY=AY-1:LOCATE AX,AY:PRINT
CHR$(214);:OX=AX:OY=AY
2460 IF AS=7 THEN PEN 2:PRINT" ";CHR$(212);:PEN 1:PRINT
CHR$(211);CHR$(210);:AX=AX+1:OX=AX+2:OY=AY
2470 IF AS=8 THEN OX=AX:OY=AY:PRINT CHR$(207);CHR$(208);:PEN
2:PRINT CHR$(209);" ";:AX=AX-1
2480 IF AS=5 THEN PEN 2:PRINT" ";:LOCATE AX,AY+1:PRINT
CHR$(206);:PEN 1:LOCATE AX,AY+2:PRINT CHR$(205);:LOCATE
AX,AY+3:PRINT CHR$(204);:AY=AY+1:OX=AX:OY=AY+2
2490 IF AS=6 THEN PRINT CHR$(201);:LOCATE AX,AY+1:PRINT
CHR$(202);:LOCATE AX,AY+2:PEN 2:PRINT CHR$(203);:PEN 1:LOCATE
AX,AY+3:PRINT" ";:OX=AX:OY=AY:AY=AY-1
2500 '
2510 IF WAVE<>4 THEN FOR I=1 TO 200-OFS*10:NEXT
2520 PEN 1
2530 RETURN
2540 INK 2,6:PEN 2:I=LEN(TEXT$):LOCATE 21-I/2,7:PRINT TEXT$;:FOR
J=1 TO 4000:NEXT:LOCATE 21-I/2,7:PRINT SPC(I);:INK 2,11,6:PEN
1:RETURN
2550 IF FB AND SCORE!>20000 THEN BASE=BASE+1:FB=0:GOSUB 2580
2560 LOCATE 6,1
2570 PEN 3:SCR$=STR$(SCORE!):LNS=LEN(SCR$)-
1:SCR$=RIGHT$(SCR$,LNS):PRINT STRING$(6-LNS,48);SCR$;:RETURN
2580 PEN 3:TEXT$=" "+STRING$(BASE-1,200):LOCATE 41-
LEN(TEXT$),25:PRINT TEXT$;:PEN 1:RETURN
2590 IF SCORE!>BEST! THEN BEST!=SCORE!:LOCATE 35,1:GOSUB 2570
2600 TEXT$="GAME OVER":GOSUB 2540
2610 IF SCORE!> HI!(9) THEN CLS:LOCATE  1,10:INK 2,6:INK 3,18:INK
0,0:BORDER 0:PEN 2:PRINT"CONGRATULATIONS!":PRINT:PEN 3:PRINT"YOU
ARE ONE OF THE TEN BEST PLAYERS":PRINT"ENTER YOUR INITIALS:" ELSE
GOTO 1150
2620 TEXT$="":I=0:LOCATE 18,16
2630 Y$=INKEY$:Y$=UPPER$(Y$):IF Y$<"A" OR Y$>"Z" THEN 2630 ELSE
I=I+1:PEN I:PRINT Y$;:TEXT$=TEXT$+Y$
2640 IF I<>3 THEN 2630 ELSE FOR I=0 TO 9:IF SCORE!>HI!(I) THEN FOR
J=9 TO I+1 STEP-1:HI!(J)=HI!(J-1):INI$(J)=INI$(J-
1):NEXT:HI!(I)=SCORE!:INI$(I)=TEXT$ ELSE NEXT
2650 GOTO 1150
```

4. L'art abstrait en BASIC

Avez-vous déjà eu la joie sans pareil de recevoir des invités qui prennent un malin plaisir à dénigrer systématiquement votre CPC AMSTRAD? Si ce n'est pas le cas, êtes-vous sûr qu'il ne vous arrivera jamais de recevoir des invités de ce style? Si ce genre d'attitude a le don de vous porter sur les nerfs, il vaudrait mieux que vous sachiez que ces gens vous poseront toujours la question à mille francs sur les possibilités graphiques et qu'ils n'auront ainsi aucun mal à vous mettre vraiment à l'épreuve. Ce n'est pas avec trois instructions PLOT et deux instructions DRAW que vous aurez une chance de réaliser une démonstration suffisamment convaincante pour démolir les arguments des critiques; il faudra tout de même leur présenter des dessins plus achevés. Mais c'est justement dans de telles situations que même le programmeur le plus chéri par les muses de l'informatique n'arrivera pas à trouver sur le champ la moindre idée suffisamment impressionnante. Et des spectateurs qui ne sont de toute façon pas objectifs seront alors définitivement convaincus que le CPC ne sait rien faire dans le domaine du graphisme!

Pour que vous soyez assuré à l'avenir de pouvoir balayer d'un revers de la main les reproches de cet ordre qui sont sans aucun fondement, ou tout simplement pour que vous puissiez exploiter pleinement de cette machine graphique qu'est le CPC tout ce qu'elle permet en BASIC, nous avons écrit ce chapitre qui vous présentera une série de programmes graphiques qui forceront chez les observateurs un cri d'étonnement.

Tous ces programmes sont imprimés sans explication et vous pouvez les compléter ou les modifier à volonté. Le résultat que les programmes doivent produire sur l'écran, si vous les tapez tels qu'ils sont imprimés, est chaque fois présenté sur une copie d'écran.

Certains de ces programmes ont besoin de temps de calcul très important avant d'achever un dessin qui ne sera plus modifié ensuite. Il est donc recommandé de stocker les dessins statiques sur disquette. Pour ce faire, il suffit d'utiliser l'instruction

```
SAVE "ECRAN",B,&C000,&4000
```

et les 16 K octets de la mémoire écran seront sauvegardés sur disquette sous forme d'un fichier binaire. Nous vous conseillons donc de placer cette instruction dans l'endroit du programme où le dessin est déjà achevé et ne sera plus modifié. Vous pouvez aussi de cette manière vous constituer une collection de dessins sur disquette que vous conserverez pour votre plaisir personnel ou pour faire taire les invités perfides.

Un dessin sauvegardé de cette façon peut ensuite être ramené sur l'écran à tout moment avec l'instruction

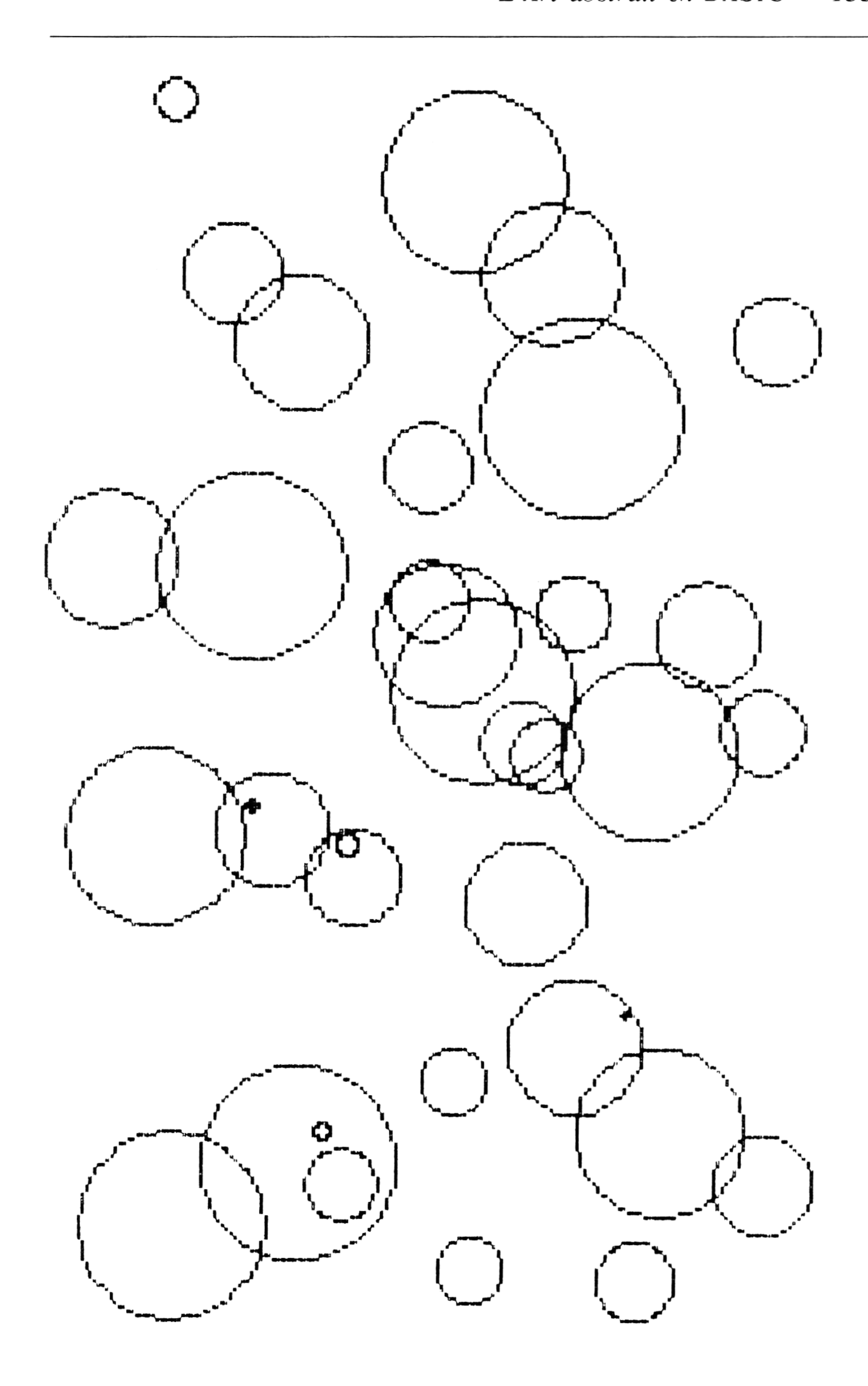
```
LOAD "ECRAN.BIN"
```

En associant plusieurs instructions LOAD, il est également facile de créer une démonstration graphique qui présentera en alternance continue les possibilités graphiques du CPC. Si vous possédez un 6128, vous pouvez encore renforcer cet effet en ayant recours au travail très simple sur plusieurs pages écran (BANKMAN).

4.1 "TOUCHE PAS A MES CERCLES !"

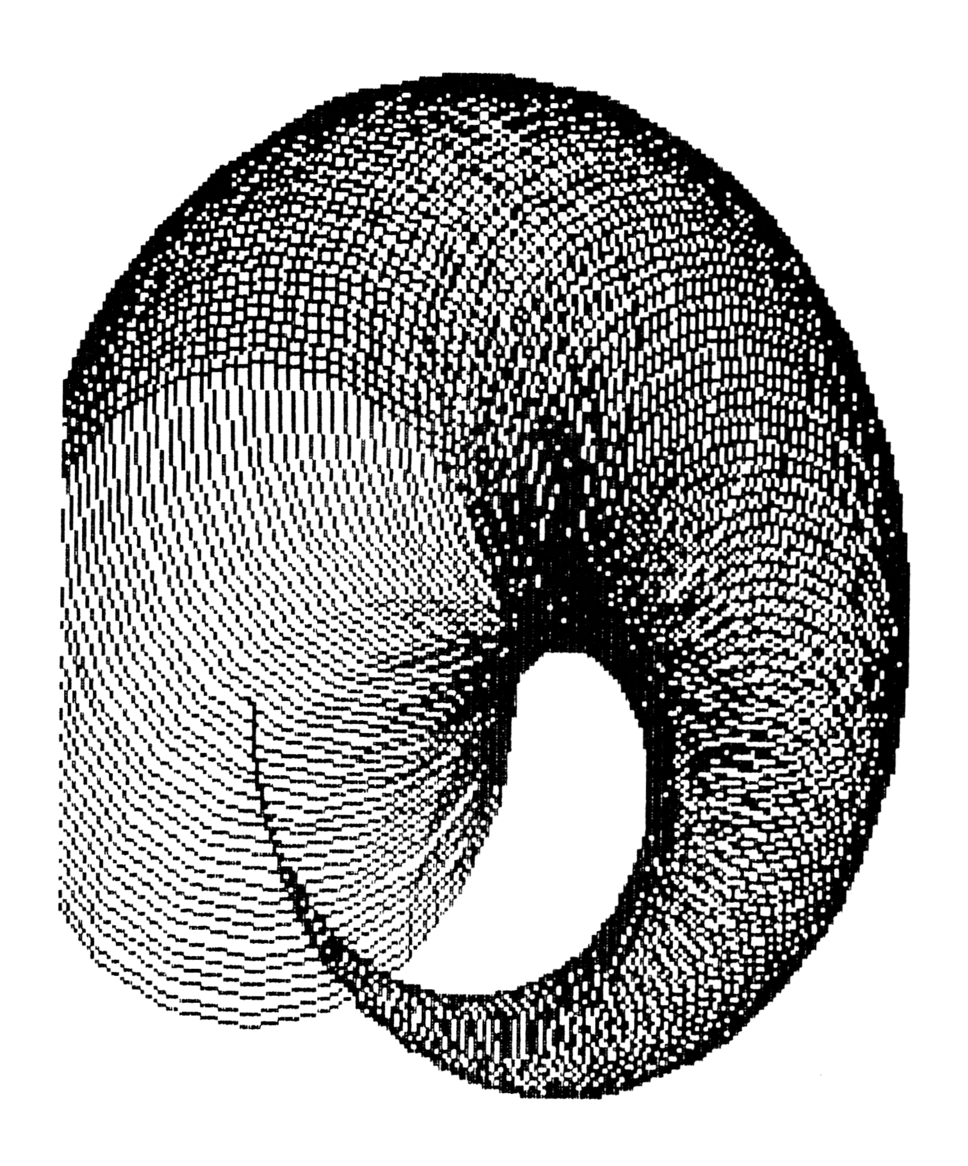
```
100 '**********************************************************
110 '***                                                    ***
120 '***           "TOUCHE PAS A MES CERCLES!"              ***
130 '***              ART ABSTRAIT EN BASIC                 ***
140 '***                                                    ***
150 '***                 JST/TAV 2.8.1986                   ***
160 '***                                                    ***
170 '**********************************************************
180 '
190 MODE 1
200 BORDER 0
210 INK 0,0
220 INK 1,26
230 '
240 DEG
250 '
260 'PRENDRE DES VALEURS ALEATOIRES POUR LE CENTRE ET LE RAYON
270 '
280 XP%=INT(RND(1)*640)
290 YP%=INT(RND(1)*400)
300 D% =INT(RND(1)*50)
310 '
320 'LE CERCLE ACTUEL PEUT-IL ETRE REPRESENTE ENTIEREMENT?
330 '
340 IF XP%+D%>639 THEN 280
350 IF YP%+D%>399 THEN 280
360 IF XP%-D%<  0 THEN 280
370 IF YP%-D%<  0 THEN 280
380 '
390 'PREMIERES COORDONNEES
400 '
410 J%=0
420 GOSUB 690
430 '
440 X1%=X2%
450 Y1%=Y2%
460 '
470 'AUTRES COORDONNEES
480 '
490 FOR J%=0 TO 15
```

```
500 GOSUB 690
510 '
520 'DESSIN DE CERCLE
530 '
540 MOVE XP%+X1%,YP%+Y1%
550 DRAW  XP%+X2%,YP%+Y2%
560 MOVE XP%+X1%,YP%-Y1%+1
570 DRAW  XP%+X2%,YP%-Y2%+1
580 MOVE XP%-X1%+1,YP%-Y1%+1
590 DRAW  XP%-X2%+1,YP%-Y2%+1
600 MOVE XP%-X1%+1,YP%+Y1%
610 DRAW  XP%-X2%+1,YP%+Y2%
620 '
630 X1%=X2%
640 Y1%=Y2%
650 '
660 NEXT
670 '
680 GOTO 280
690 '
700 'CALCUL DE COORDONNEES
710 '
720 X2%=INT(COS(J%*6)*D%+0.5)
730 Y2%=INT(SIN(J%*6)*D%+0.5)
740 '
750 RETURN
```



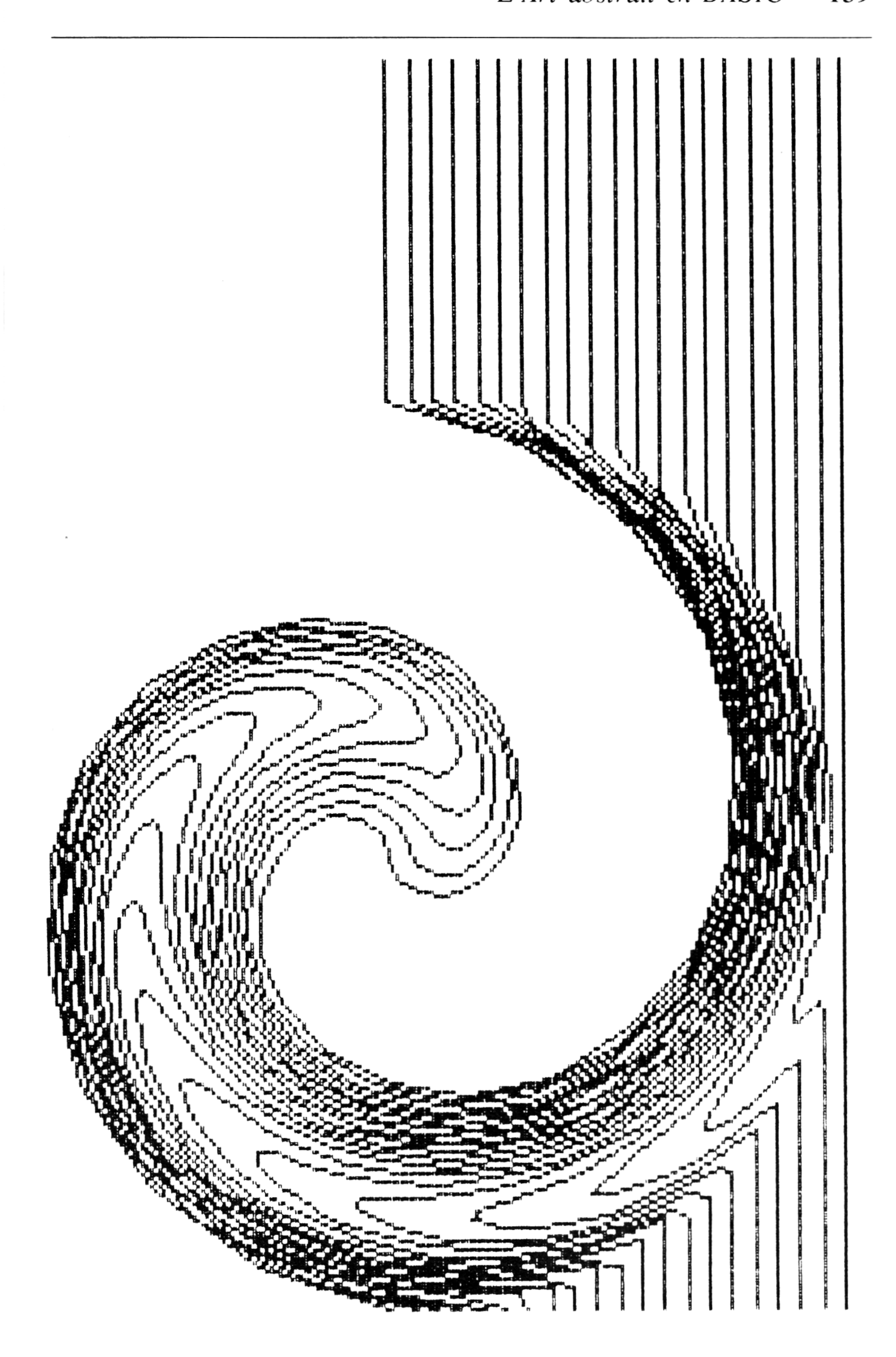
4.2 CORNE

```
100 '***************************************************************
110 '***                                                         ***
120 '***    CORNE - ART ABSTRAIT EN BASIC    JST 21.6.1986       ***
130 '***                                                         ***
140 '***************************************************************
150 '
160 MODE 1
170 '
180 BORDER 0
190 INK 0,0
200 INK 1,26
210 '
220 FOR I%=0 TO 3000
230 '
240 A=I%*PI/3000
250 B=120*(0.5+COS(A)/2)
260 C=A*2
270 D=C*150
280 E=B*COS(D)
290 F=B*SIN(D)
300 G=240+120*SIN(C)+F
310 X%=INT(1.3*G)
320 Y%=INT(240+120*COS(C)+E)
330 Y%=Y%-50
340 '
350 IF I%=0 THEN MOVE X%,Y%
360 IF I%>0 THEN DRAW X%,Y%
370 '
380 NEXT I%
390 '
400 GOTO 400
```



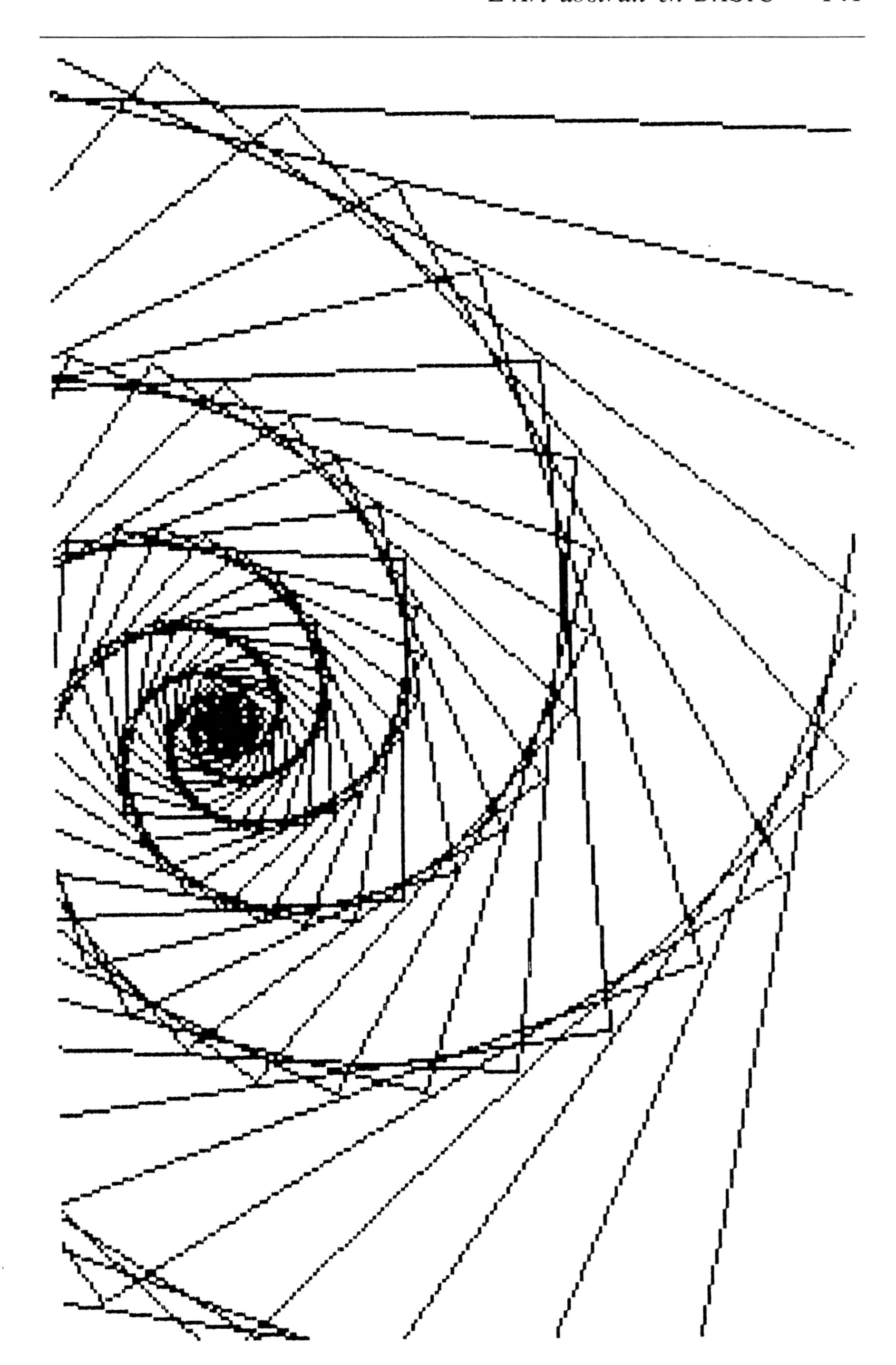
4.3 VAGUE

```
100 '**************************************************************
110 '***                                                        ***
120 '***   VAGUE - ART ABSTRAIT EN BASIC   JST 21.6.1986        ***
130 '***                                                        ***
140 '**************************************************************
150 '
160 MODE 1
170 '
180 BORDER 0
190 INK 0,0
200 INK 1,26
210 '
220 FOR I%=0 TO 20
230 FOR J%=0 TO 55
240 '
250 X=I%/20-1
260 Y=J%/20-1
270 D=SQR(X*X+Y*Y)
280 IF X<>0 THEN A=ATN(Y/X)
290 IF X=0 THEN A=PI/2*SGN(Y)
300 IF X<0 THEN A=A+PI
310 IF D<1 THEN A=A+PI*2*(1-D)
320 B=D*SIN(A)
330 C=D*COS(A)
340 E=1+0.95*B
350 F=1+0.95*C
360 X%=INT(449/2*E)
370 X%=X%+15
380 Y%=INT(449/2*F)
390 IF J%=0 THEN MOVE X%,Y%
400 IF J%>0 THEN DRAW X%,Y%
410 '
420 NEXT J%
430 NEXT I%
440 '
450 GOTO 450
```



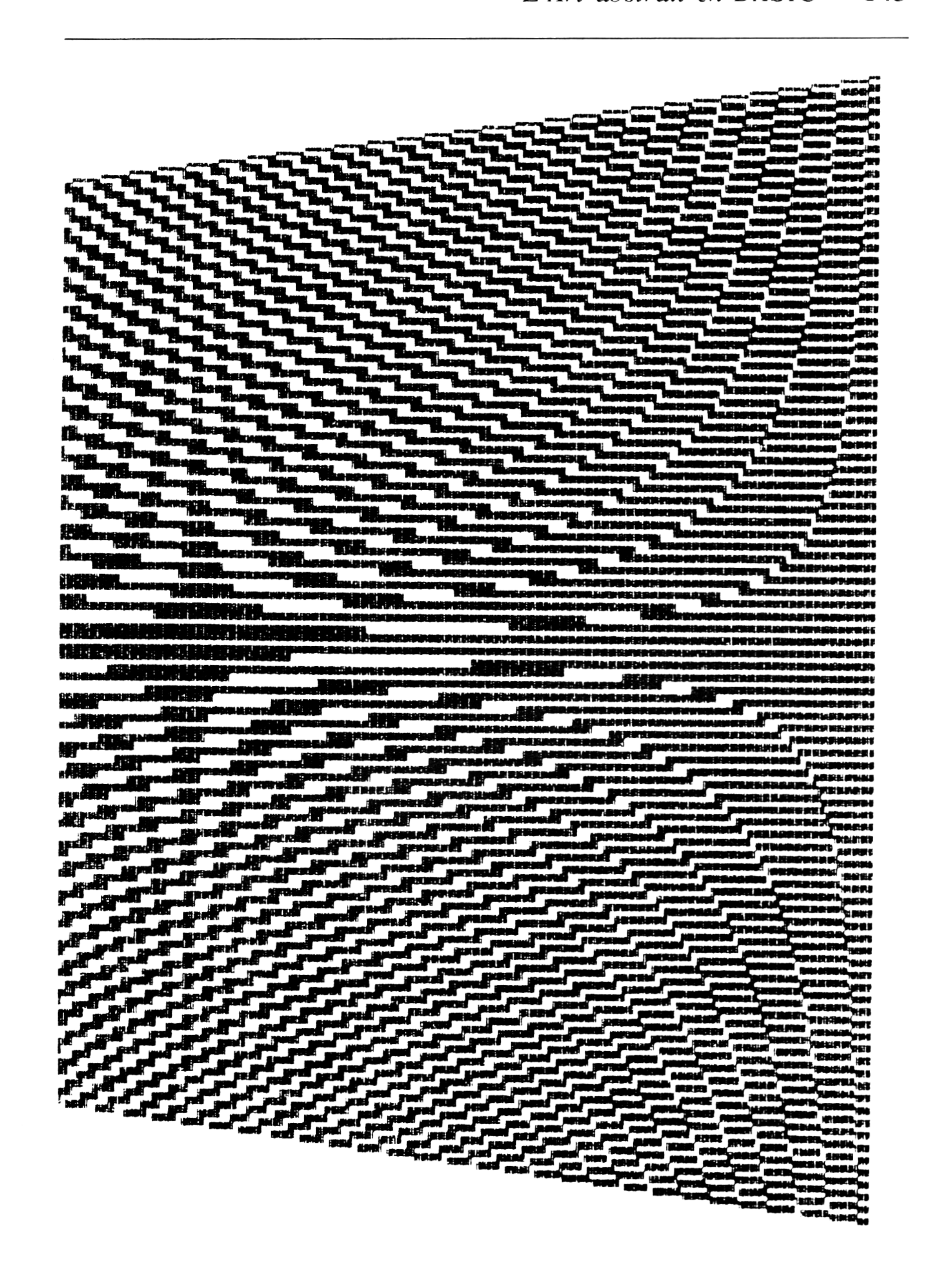
4.3 PROFONDEUR

```
100 '******************************************************
110 '**                                                  **
120 '** PROFONDEUR - ART ABSTRAIT EN BASIC JST 21.6.1986 **
130 '**                                                  **
140 '******************************************************
150 '
160 MODE 1
170 '
180 INK 0,0
190 INK 1,26
200 BORDER 0
210 '
220 A=0
230 B=600
240 '
250 FOR i=0 TO 300
260 '
270 X=X+B*COS(A)
280 Y=Y+B*SIN(A)
290 X%=INT(X)
300 Y%=INT(Y)
310 '
320 IF I=0 THEN MOVE X%,Y%
330 IF I>0 THEN DRAW X%,Y%
340 '
350 A=A+1.52
360 B=B*0.98
370 '
380 NEXT I
390 '
400 GOTO 400
```



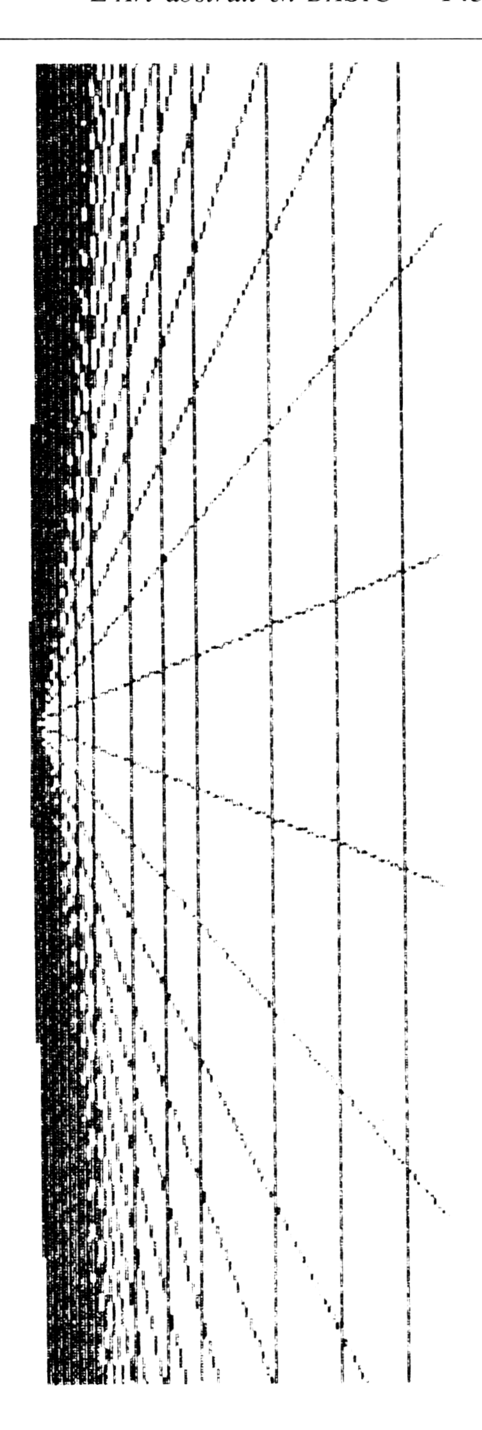
4.4 MODELE D'INTERFERENCES

```
100 '************************************************************
110 '***                                                      ***
120 '*** MODELE D'INTERFERENCES - Art Abstrait en BASIC       ***
130 '***                    JST 14.6.1986                     ***
140 '***                                                      ***
150 '************************************************************
160 '
170 MODE 1
180 '
190 n=0
200 x1=40
210 y1=0
220 x2=600
230 y2=4000
240 '
250 x1=x1+3
260 x2=x2-3
270 '
280 IF x1>600 THEN x1=41:n=n+1
290 IF x2<40 THEN x2=599
300 '
310 PLOT x1,y1,1
320 PRINT CHR$(23);CHR$(1);
330 DRAW x2,y2,1
340 '
350 IF n=1 THEN n=2
360 IF x1=41 THEN INK 1,n
370 '
380 GOTO 250
```



4.5 GRIDRUNNER

```
100 '*******************************************************
110 '***                                                ***
120 '***                 GRIDRUNNER                     ***
130 '***                TAV 20.10.1986                  ***
140 '***                                                ***
150 '*******************************************************
160 '
170 '
180 MODE 0
190 BORDER 0
200 INK 0,0
210 INK 1,9
220 FOR I=2 TO 15
230 INK I,0
240 NEXT
250 FOR I=2 TO 15
260 J=196-I*2
270 FOR K=1 TO 3
280 MOVE 0,J,I
290 DRAW 639,J,I
300 J=J-28*K-(I*2-4)*K
310 NEXT
320 NEXT
330 FOR I=-10000 TO 10000 STEP 160
340 MOVE 320,200
350 DRAW I,0,1
360 NEXT
370 FOR I=2 TO 15
380 INK I,9
390 CALL &BD19
400 INK I,0
410 NEXT
420 GOTO 370
```



5. Un bon croquis vaut mieux qu'un long discours

Les graphiques d'entreprise

Essayez donc de vous mettre à la place du patron d'un groupe industriel qui voudrait faire comprendre aux responsables de ses filiales que l'entreprise ne fait plus un chiffre d'affaires suffisant et que des efforts importants doivent donc être consentis pour que le chiffre d'affaires recommence à progresser. Pour faire cette communication, il a convoqué une réunion de tous les responsables de ses filiales. Ces Messieurs sont maintenant réunis dans l'attente d'un évènement important et le grand patron commence son exposé sur la situation du chiffre d'affaires de l'entreprise : "Notre chiffre d'affaires a été de 13,53 milliards de francs en 1975, en 1976 il a progressé de 3,4 milliards pour atteindre finalement en 1977 19 milliards..."

L'exposé peut durer encore un certain temps mais même les auditeurs les mieux disposés auront déjà perdu le fil à partir de ce moment. Ils "décrocheront" et se laisseront emporter par leurs pensées vers des univers lointains.

Il est certainement difficile à admettre pour un homme d'affaires sérieux que ses développements sur les chiffres d'affaires soient impossibles à suivre. Mais que peut-il faire d'autre? Il dispose au fond de plusieurs possibilités mais il a l'une d'entre elles sous la main ou, plus exactement, devant lui sur son bureau. Comme il ne peut naturellement pas, lui non plus, avoir en tête les chiffres d'affaires de toutes les années antérieures, il a demandé à son service comptable de regrouper ces chiffres dans un tableau.

S'il a pris le soin de faire reproduire ce tableau et de le distribuer à ses auditeurs, cela peut déjà le dispenser de nombreux développements et cela rendra la masse des données plus facile à appréhender. Chaque auditeur pourra se baser sur le niveau d'une année déterminée et l'exposé pourra se limiter à décrire les tendances et les évolutions.

Cette façon de représenter les chiffres sous forme de tableau a cependant aussi ses inconvénients.

Les chiffres ont quelque chose d'aride et d'abstrait qui ne permet pas facilement d'en dégager une impression concrète. S'il s'agit en plus de comparer différents chiffres, il devient encore plus ardu d'obtenir une présentation claire.

Chiffres d'affaires de 1975 à 1985

1975 :	13,53 Milliards F
1976 :	16,93 Milliards F
1977 :	18,97 Milliards F
1978 :	23,21 Milliards F
1979 :	27,47 Milliards F
1980 :	28,90 Milliards F
1981 :	31,23 Milliards F
1982 :	32,31 Milliards F
1983 :	32,98 Milliards F
1984 :	30,26 Milliards F
1985 :	27,45 Milliards F

Ce tableau montre bien sûr clairement que le chiffre d'affaires a baissé au cours des deux dernières années mais cette situation apparaîtrait de façon beaucoup plus évidente si on représentait ces statistiques dans un graphique. On pourrait par exemple dessiner un histogramme pour chacune des années 1975 à 1985, la longueur de chaque histogramme étant déterminée par le nombre plus ou moins élevé qu'il représente. La figure 7 montre comment ces montants pourraient être représentés graphiquement à l'aide d'histogrammes.

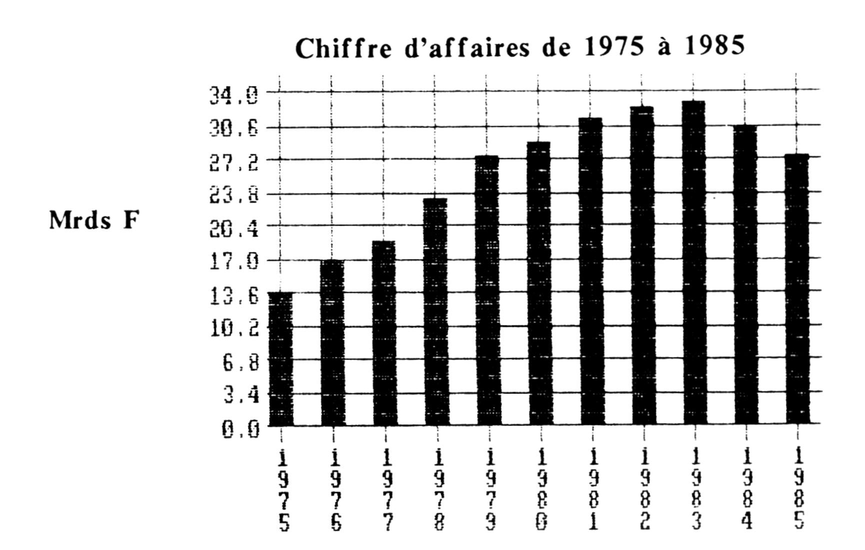


Figure 7 : **Représentation sous forme d'histogrammes**

Le rapprochement des différents histogrammes permet de lire immédiatement l'évolution du chiffre d'affaires et le grand patron dispose ainsi d'un point d'appui solide pour son argumentation. Il pourra ainsi convaincre les derniers incrédules du fait que les statistiques de la société ne sont pas au beau fixe.

Il existe bien d'autres façons de convertir des colonnes de chiffres en graphiques parlants mais ce seul exemple des histogrammes illustre déjà clairement combien un graphique est plus facile à lire que des chiffres nus. Ce principe ne s'applique d'ailleurs pas uniquement au monde des affaires.

5.1 LE GRAPHIQUE DE POINTS

Voici maintenant une première forme de graphique d'entreprise qui consiste à marquer par des points les emplacements où se situent les valeurs à représenter. Nous avons choisi de commencer nos explications par ce graphique parce qu'il ne nécessite pas un

travail trop difficile pour convertir les statistiques en un graphique. Ce graphique peut en effet être réalisé sans préparation particulière contrairement aux autres types de graphiques que nous vous présenterons ensuite.

La structure d'un graphique d'entreprise est toujours la même, elle repose sur deux axes qui se croisent en angle droit. Un des axes comporte une graduation qui indique la provenance des statistiques alors que l'autre axe est gradué d'après des unités de grandeur. Le premier axe peut être gradué d'après les années, les mois, les filiales ou tout autre notion de ce type. Le second axe représente les quantités, les montants ou d'autres valeurs de ce type. Les statistiques qui doivent être représentées déterminent à cet égard la forme de la graduation. Pour ne pas rester trop longtemps dans la théorie, prenons l'exemple d'une personne privée qui veut se faire une idée des kilomètres parcourus par son véhicule. Il a pour cela réalisé un tableau montrant les kilomètres parcourus pour chaque mois.

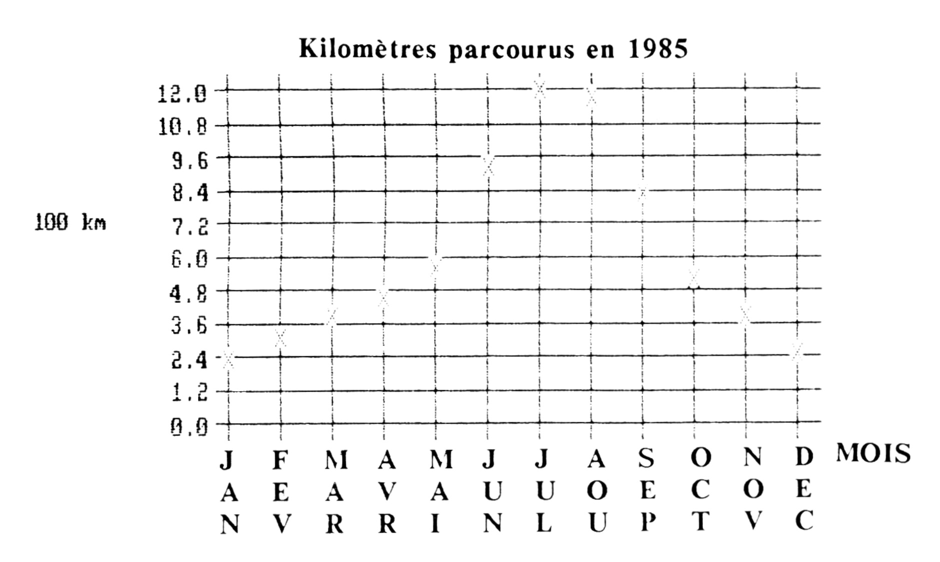
Kilomètres parcourus en 1985

Janvier	:	230 km
Février	:	310 km
Mars	:	390 km
Avril	:	450 km
Mai	:	560 km
Juin	:	920 km
Juillet	:	1200 km
Août	:	1170 km
Septembre	:	830 km
Octobre	:	520 km
Novembre	:	390 km
Décembre	:	250 km

Pour représenter maintenant ces statistiques dans un graphique, il nous faut tout d'abord graduer les deux axes utilisés. La graduation du premier axe correspondra aux douze mois de l'année 1985. Le second axe devra être gradué en kilomètres. Il faudra choisir la graduation la plus élevée d'après le nombre maximum de kilomètres parcourus en un mois, dans notre exemple donc 1200 km.

Comme toutes les distances à représenter se situent dans un ordre de grandeur de quelques centaines de kilomètres, la graduation peut se faire par unités de 100 km.

Une fois le graphique ainsi préparé, il sera facile de réaliser le marquage des points. Le tableau nous indique les valeurs correspondant aux différents mois de sorte que nous pouvons marquer pour chaque mois la graduation de l'axe des kilomètres où se situe la valeur correspondante. Ce travail sera effectué pour tous les mois de Janvier à Décembre et le graphique sera terminé. Pour qu'il puisse vraiment être utilisé plus tard, il faut encore que le graphique comporte des inscriptions explicatives. Il faut donc inscrire un titre qui indique de quel type de graphique il s'agit ainsi que des commentaires sur les deux axes.



Copie d'écran 7 : **Graphique de points**

Le graphique de points ci-dessus a été réalisé grâce à un programme qui vous mâche tout le travail de réalisation du graphique. Vous avez simplement besoin d'un tableau contenant les valeurs que vous pourrez entrer dans le programme. Le programme se charge du dessin des axes, de l'écriture des commentaires et de la conversion graphique des valeurs.

Après le lancement du programme, on demande à l'utilisateur quel titre le graphique devra recevoir, dans notre exemple le titre était "Kilomètres parcourus en 1985", et quelle désignation est prévue pour l'échelle des valeurs. Ces indications faites, le programme demande à l'utilisateur les valeurs numériques pour les différents mois à représenter (les valeurs entrées ne doivent pas être inférieures à zéro).

Avant que le programme ne commence à dessiner le graphique, la graduation de l'échelle des valeurs est automatiquement calculée d'après la valeur maximale. Cette graduation automatique peut parfois déboucher sur des valeurs vraiment "biscornues" mais c'est le prix de l'automatisation de cette opération.

Après que le programme ait dessiné les deux axes et leurs graduations qui traversent tout le graphique pour le rendre plus clair, le programme commence à entrer les valeurs dans le graphique. A cet effet, les points situés à l'intersection des mois et des valeurs correspondantes sont marqués par des croix. La valeur concernée est alors désignée par le centre de la croix.

```
100 MODE 2
110 BORDER 0
120 INK 0,0
130 INK 1,11
131 SYMBOL 255,0,65,34,20,8,20,34,65
140 DIM MY(12)
150 MA=0
160 '
170 'SORTIE DE L'EN-TETE DU PROGRAMME
180 '
190 LOCATE 1,3
200 PRINT TAB(33);"GRAPHIQUE DE POINTS"
210 PRINT TAB(33);"JST  28.7.1986"
220 MOVE 16,383
230 DRAW 16,320
240 DRAW 623,320
250 DRAW 623,383
260 DRAW 16,383
270 '
280 'ENTRER LES INSCRIPTIONS
290 '
```

```
300 LOCATE 36,8
310 PRINT"|--------------------------------------|"
320 LOCATE 1,8
330 INPUT"TITRE (40 LETTRES MAXI)";UE$
340 LOCATE 36,10
350 PRINT"|--------|"
360 LOCATE 1,10
370 INPUT"VALEURS (10 LETTRES MAXI)";WE$
380 '
390 'ENTRER LES DIFFERENTES VALEURS ET CALCULER LE MAXIMUM
400 '
410 PRINT
420 PRINT"ENTREZ MAINTENANT LES VALEURS POUR LES DIFFERENTS MOIS!"
430 PRINT
440 '
450 FOR I=1 TO 12
460 READ MOIS$
470 PRINT USING "\        \";MOIS$;
480 INPUT MY(I)
490 IF MA<MY(I) THEN MA=MY(I)
500 NEXT I
510 '
520 CLS
530 '
540 'CALCULER LA POSITION DE L'INSCRIPTION
550 '
560 T1=LEN(UE$)
570 T2=LEN(WE$)
580 PU=40-(T1/2)
590 PW=6-(T2/2)
600 '
610 'SORTIR L'INSCRIPTION
620 '
630 LOCATE 69,20
640 PRINT "MOIS"
650 LOCATE PU,1
660 PRINT UE$
670 LOCATE PW,9
680 PRINT WE$
690 '
700 'DESSINER LE SYSTEME DE COORDONNEES
```

```
710 '
720 MOVE 163,375
730 DRAW 163,112
740 DRAW 530,112
750 '
760 'GRADUATION DE L'AXE DES X
770 '
780 FOR X=163 TO 515 STEP 32
790 MOVE X,100
800 DRAW X,375
810 NEXT X
820 '
830 'TITRE DE L'AXE DES X
840 '
850 FOR MY=1 TO 3
860 MY1=MY+19
870 RESTORE
880 FOR M=1 TO 12
890 MX=M*4+17
900 READ MOIS$
910 MOIS$=RIGHT$(LEFT$(MOIS$,MY),1)
920 LOCATE MX,MY1
930 PRINT MOIS$
940 NEXT M
950 NEXT MY
960 '
970 'GRADUATION DE L'AXE DES Y
980 '
990 FOR Y=362 TO 112 STEP -25
1000 MOVE 155,Y
1010 DRAW 530,Y
1020 NEXT Y
1030 '
1040 'TITRE DE L'AXE DES Y
1050 '
1060 IF INT(MA)<>MA THEN MA=INT(MA)+1
1070 IF INT(MA/2)<>MA/2 THEN MA=MA+1
1080 '
1090 TAG
1100 '
1110 FOR I=0 TO 10
1120 PY=25*I+117
```

```
1130 MOVE 95,PY
1140 PRINT USING "#####.#";MA/10*I;
1150 NEXT I
1160 '
1170 'FIXER LES POINTS
1180 '
1190 F=131
1200 FOR I=1 TO 12
1210 MY=((MY(I)/MA)*250)+112
1220 F=F+32
1240 MY=MY+8
1280 MOVE F-4,MY
1290 PRINT CHR$(255);
1300 NEXT I
1310 '
1320 GOTO 1320
1330 '
1340 'DATAs AVEC LES NOMS DE MOIS
1350 '
1360 DATA JANVIER,FEVRIER,MARS,AVRIL,MAI,JUIN
1370 DATA JUILLET,AOUT,SEPTEMBRE,OCTOBRE,NOVEMBRE,DECEMBRE
```

Description du programme :

100-150 Définitions, dimensionnements et pré-affectations. On fixe le MODE 2 et une combinaison de couleurs avec beaucoup de contraste. Un nouveau caractère est défini sous le code de caractère 255. Il a l'apparence d'une croix comme celles qu'on inscrit sur un bulletin de loto. Ce caractère convient parfaitement pour marquer un point bien précis. Dans le programme "Graphique de points", il désignera les endroits de l'échelle où figurent les valeurs correspondant aux différents mois.

La variable indicée MY est dimensionnée à 13 éléments. Sous les indices 1 à 12 seront stockées les valeurs pour les douze mois. La variable MA, qui servira à conserver dans le déroulement du programme la valeur maximale pour les mois, est fixée sur 0.

160-260 *Sortie de l'en-tête du programme*
Le nom et le titre du programme sont sortis sur l'écran et encadrés.

270-370 *Lecture des titres*
Le titre et le commentaire de l'échelle des valeurs du graphique à réaliser sont entrés à l'aide de l'instruction INPUT. La place réservée pour les entrées est marquée sur l'écran. Les entrées qui sortent du cadre représenté ne sont pas corrigées.

380-520 *Entrer les différentes valeurs et déterminer la valeur maximale*
Après un message indiquant qu'il s'agit maintenant d'entrer les valeurs pour les différents mois, le programme commence le traitement d'une boucle FOR-TO-NEXT. Dans cette boucle, qui est parcourue exactement 12 fois, on demande à l'utilisateur d'effectuer les entrées correspondant aux différents mois. A cet effet, la variable MOIS$ se voit attribuer lors de chaque parcours de la boucle un des noms de mois placés dans le bloc de DATA à la fin du programme.

Après que le nom du mois ait été sorti sous un format uniforme pour tous les mois, le programme attend la valeur qui servira de base à la représentation graphique du mois correspondant. La ligne 490 compare le contenu actuel de MY(I) à celui de MA. Si MA contenait la valeur la plus petite, elle se voit affecter la valeur la plus grande des deux. Après que la boucle FOR-TO-NEXT ait été parcourue entièrement, MA contiendra donc la valeur maximale des douze mois.

Après que toutes les indications nécessaires aient été collectées, l'écran est vidé pour réaliser le graphique.

530-590 *Calculer la position pour le titre*
La longueur du titre et du commentaire de l'échelle de valeurs est déterminée à l'aide de la fonction LEN et elle est affectée aux variables T1 et T2. Le contenu de ces variables divisé par 2 est chaque fois

soustrait de la position X qui devra coïncider par la suite avec le milieu de l'inscription. On peut ainsi en déduire la position X du curseur de texte à laquelle devra commencer la sortie des textes pour qu'ils apparaissent centrés dans l'emplacement voulu. Les coordonnées X pour le début des sorties de texte se trouvent dans les variables PU (titre) et PW (échelle de valeurs).

600-680 *Sortir le commentaire*
Le commentaire pour le graphique est sorti sur l'écran en utilisant les positions de sortie de texte calculées.

690-740 *Dessiner le système de coordonnées*
Le système de coordonnées sur lequel reposera le graphique est dessiné avec une instruction MOVE et deux instructions DRAW.

750-810 *Graduation de l'axe des X*
Après que cette partie du programme ait été exécutée, le système de coordonnées se trouve divisé par 12 lignes verticales partant de l'axe des X. Cette subdivision fondée sur les 12 mois est effectuée pour rendre le graphique plus lisible.

820-950 *Commentaires de l'axe des X*
Cette partie du programme écrit les trois premières lettres des noms de mois l'une sous l'autre verticalement dans les emplacements prévus à cet effet sur l'axe des X. Ce travail est effectué par deux boucles FOR-TO-NEXT imbriquées.

La boucle extérieure est parcourue exactement trois fois et elle traite chaque fois une des trois premières lettres des noms de mois. La variable de comptage de cette boucle (MY) permet en ligne 860 de déterminer la position Y pour la sortie des lettres correspondant au parcours actuel de la boucle. Le pointeur DATA est réinitialisé avant l'entrée dans la boucle FOR-TO-NEXT intérieure. Cela permet de relire les noms de mois lors de chaque parcours de boucle.

Chacun des 12 parcours de la boucle intérieure traite un mois déterminé. La boucle calcule d'abord d'après la variable de comptage M la coordonnée X pour la sortie de la lettre. Cette coordonnée est affectée à MX. La position de la sortie de texte figure ainsi dans les variables MX et MY1. Les lignes 900 et 910 déterminent quelle lettre doit être sortie.

La ligne de programme 910 isole la lettre voulue du nom de mois tiré du bloc de DATA. On détache pour cela, en fonction de la variable de comptage MY, la première lettre, les deux premières lettres ou les trois premières lettres du nom du mois (LEFT$(MOIS$,MY)). On réserve chaque fois la dernière de ces lettres pour la sortie. Le curseur de texte est alors positionné dans l'emplacement calculé et la lettre isolée qui se trouve dans la variable MOIS$ est sortie.

960-1020 *Graduation de l'axe des Y*
Cette partie du programme divise le système de coordonnées par 11 lignes horizontales partant de l'axe des Y. La ligne la plus basse (l'axe des X) représente le point zéro sur l'échelle des valeurs. Toutes les autres divisions servent à obtenir une attribution plus fine des indications de l'échelle aux points du graphique. C'est pourquoi elles coupent tout le graphique comme les divisions verticales.

1030-1150 *Commentaires sur l'axe des Y*
Le premier travail de cette partie du programme est de déterminer la valeur maximale (MA) résultant des valeurs entrées pour les différents mois. La ligne 1060 élimine les décimales éventuelles et augmente la valeur maximale de un si ce n'était pas une valeur entière. Cela est indispensable pour qu'on ne puisse pas aboutir à des valeurs supérieures à la graduation la plus élevée de l'échelle. Si la valeur maximale est un nombre impair, elle est augmentée de un en ligne 1070 pour éviter une subdivision trop complexe de l'échelle des valeurs.

Après que la sortie de texte ait été reliée au curseur graphique (TAG), une boucle FOR-TO-NEXT écrit les commentaires sur l'axe des Y. Lors de chacun des 11 parcours de la boucle, on calcule d'abord dans quelle position le texte pour le parcours actuel devra être sorti. Cette position est affectée à la variable PY. L'inscription se fait de bas en haut. C'est chaque fois la valeur de la variable de comptage multipliée par le dixième de la valeur maximale qui est sortie.

1160-1320 *Fixer les points*
Une boucle FOR-TO-NEXT dessine les croix qui constituent le graphique proprement dit. La ligne 1210 calcule d'abord la position Y pour le centre de la croix à partir de la valeur correspondant au mois désigné par la variable de comptage. Cette valeur pour le mois est calculée à partir de la valeur maximale et de deux constantes.

La position X de la croix est obtenue en ajoutant 32, lors de chaque parcours de la boucle, à la variable F qui a été fixée sur la valeur défaut 131 avant la boucle. Les deux coordonnées calculées doivent être converties de façon à obtenir les coordonnées du coin supérieur gauche de la matrice de caractère puisqu'il ne s'agit pas de sortir un point mais bien un caractère.

Cela est fait en ligne 1240 pour la position Y. La position X est adaptée lors du déplacement du curseur graphique (ligne 1280) car elle ne doit pas être réécrite. La croix définie sous le code de caractère 255 est placée sur l'écran dans la position du curseur graphique ainsi obtenue.

La valeur correspondant à chaque mois est marquée par l'intersection des deux lignes de la croix.

La ligne 1320 contient une boucle sans fin qui a pour simple but de bloquer le programme pour que le graphique ne soit pas détruit par le message "Ready".

1330-Fin *DATAs avec les noms de mois*
C'est ici que figurent, sur deux lignes de DATA, les noms des douze mois qui sont utilisés par les parties ENTRER LES DIFFERENTES VALEURS ET CALCULER LA VALEUR MAXIMALE et COMMENTAIRE DE L'AXE DES X.

5.2 CPC-CHART - LE GENERATEUR DE GRAPHIQUE

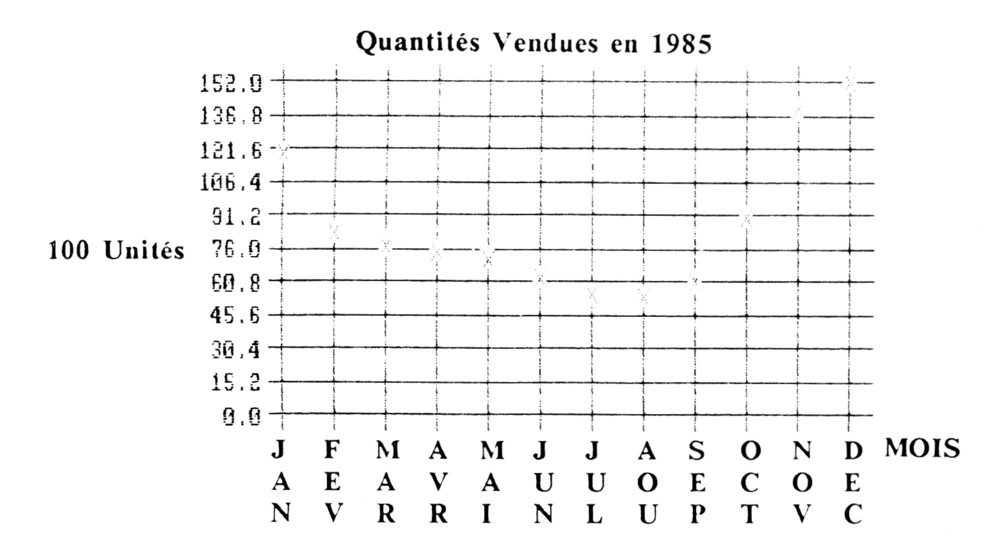
La représentation de points sur un graphique de points constitue une présentation objective car elle ne livre vraiment que les faits. D'autres formes de représentation permettent cependant non seulement de présenter les faits mais de proposer une évaluation des statistiques présentées. On peut ainsi exercer une certaine influence sur ceux qui examinent le graphique. C'est le cas par exemple des courbes qui sont obtenues en reliant les différentes valeurs présentées sur le graphique par un trait. Les courbes ainsi réalisées permettent de souligner de façon particulièrement nette les tendances qui se dégagent des statistiques étudiées.

Une autre forme de représentation est constituée par les histogrammes dont nous parlions dans notre introduction sur les graphiques d'entreprise. Les histogrammes sont des barres qu'on dessine de la valeur calculée à l'axe indiquant la provenance des statistiques. L'utilisation des barres donne artificiellement un poids particulier aux faits représentés.

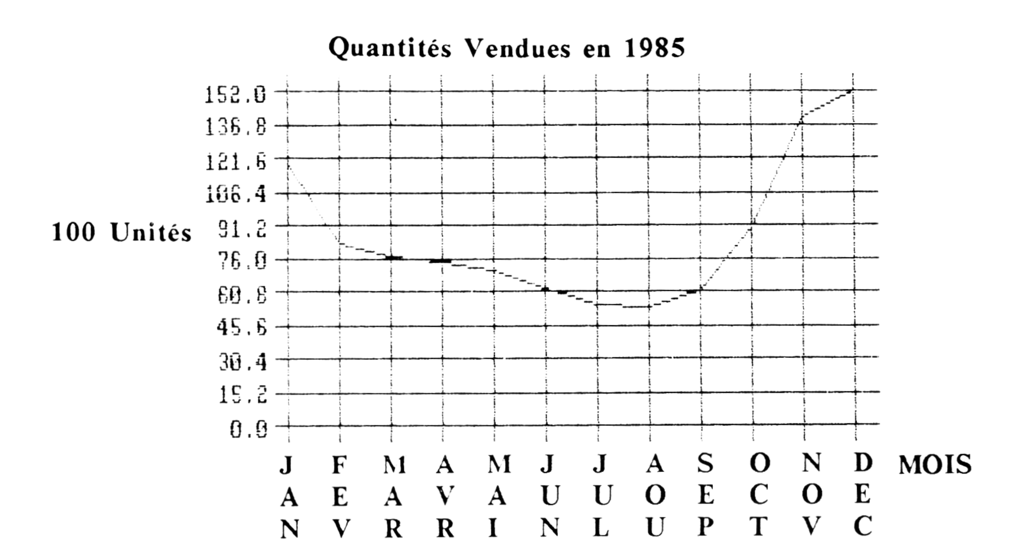
Mais vous pourrez juger par vous-même de la différence dans l'effet produit selon qu'on utilise telle ou telle forme de représentation pour les mêmes statistiques de base. Les trois graphiques suivants (points, courbe et histogrammes) reposeront sur une même base, une collection de statistiques de vente. Il s'agit de quantités vendues pour les différents mois de 1985 d'un produit imaginaire.

Quantités vendues pour l'année 1985

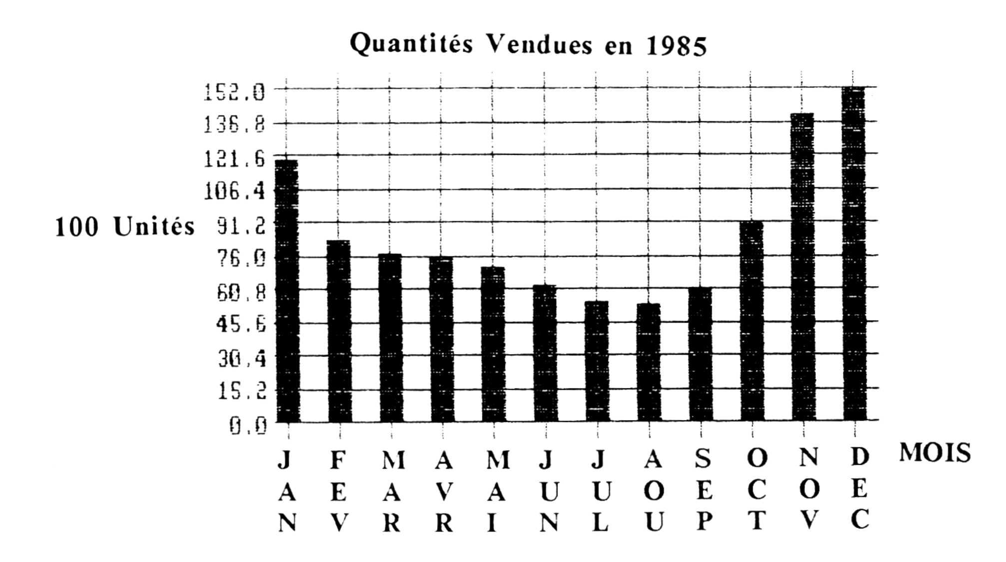
Janvier	:	12000 unités
Février	:	8300 unités
Mars	:	7700 unités
Avril	:	7400 unités
Mai	:	7100 unités
Juin	:	6200 unités
Juillet	:	5500 unités
Aout	:	5400 unités
Septembre	:	6100 unités
Octobre	:	9000 unités
Novembre	:	14000 unités
Décembre	:	15200 unités



<u>Copie d'écran 8 :</u> **Les graphiques de points marquent les faits**



Copie d'écran 9 : Une courbe souligne les tendances



Copie d'écran 10 : Les histogrammes donnent un poids particulier

Pour que vous puissiez, vous aussi, disposer à l'avenir d'un outil vous permettant d'illustrer vos statistiques privées ou professionnelles à l'aide de ces trois formes de graphique, nous vous proposons le programme "CPC-Chart". Dans la version que nous vous présentons, ce programme est conçu pour représenter des valeurs se rapportant aux différents mois d'une année. Vous pourrez cependant très facilement l'adapter à d'autres situations grâce à la description détaillée du programme que nous vous fournissons.

Le mode d'emploi de "CPC-Chart" est semblable dans les grandes lignes à celui du programme "graphique de points" que nous connaissons déjà. Ici aussi, il faut d'abord entrer les commentaires devant figurer sur le graphique ainsi que l'échelle de valeurs. On entre ensuite les valeurs pour les différents mois. Les valeurs entrées ne doivent pas ici non plus être inférieures à zéro. C'est ensuite que le mode d'emploi du programme se distingue de celui du "graphique de points". On demande en effet maintenant à l'utilisateur de choisir une forme de représentation des données. Il peut choisir entre des graphiques de points, de courbes et d'histogrammes. Une fois le type de représentation sélectionné, "CPC-Chart" dessine le graphique. Toutes les divisions et graduations sont ici aussi effectuées automatiquement.

Une fois que le graphique a été dessiné sur l'écran, l'utilisateur rencontre un second menu à l'intérieur du graphique. Il peut choisir ici s'il veut faire imprimer le graphique réalisé sur papier (copie d'écran), s'il veut revenir au menu de choix d'une forme de représentation ou s'il veut sortir du programme. Si l'utilisateur a choisi "copie d'écran", le menu disparaît de l'écran et une réplique exacte du graphique est réalisée sur l'imprimante.

```
1000 '*********************************************************
1010 '***                                                   ***
1020 '***      CPC-CHART - GENERATEUR DE GRAPHIQUES         ***
1030 '***                 JST  30.7.1986                    ***
1040 '***                                                   ***
1050 '*********************************************************
1060 '
1070 MODE 2
1080 MEMORY &8000
```

```
1090 BORDER 0
1100 INK 0,0
1110 INK 1,11
1120 DIM MY(12)
1130 MA=0
1140 '
1150 'COPIE DU GRAPHIQUE SUR IMPRIMANTE
1160 '
1170 FOR A=&A000 TO &A0BF
1180 READ D
1190 POKE A,D
1200 Z=Z+D
1210 NEXT A
1220 '
1230 DATA &cd,&ba,&bb,&cd,&e7,&bb,&32,&bd
1240 DATA &a0,&cd,&6c,&a0,&21,&8f,&01,&22
1250 DATA &be,&a0,&11,&00,&00,&3e,&07,&32
1260 DATA &c0,&a0,&cd,&7c,&a0,&0e,&00,&3a
1270 DATA &c0,&a0,&47,&e5,&d5,&c5,&cd,&f0
1280 DATA &bb,&c1,&d1,&21,&bd,&a0,&be,&e1
1290 DATA &37,&20,&01,&a7,&cb,&11,&2b,&2b
1300 DATA &10,&e9,&cd,&af,&a0,&79,&cd,&a6
1310 DATA &a0,&13,&e5,&21,&7f,&02,&37,&ed
1320 DATA &52,&e1,&38,&05,&2a,&be,&a0,&18
1330 DATA &cc,&23,&7c,&b5,&c8,&2b,&11,&00
1340 DATA &00,&22,&be,&a0,&3e,&07,&bd,&20
1350 DATA &b9,&7c,&b4,&20,&b5,&3e,&04,&32
1360 DATA &c0,&a0,&18,&ae,&3e,&1b,&cd,&a6
1370 DATA &a0,&3e,&31,&cd,&a6,&a0,&00,&00
1380 DATA &00,&00,&00,&c9,&e5,&3e,&42,&cd
1390 DATA &1e,&bb,&e1,&28,&02,&e1,&c9,&3e
1400 DATA &0d,&cd,&a6,&a0,&3e,&0a,&cd,&a6
1410 DATA &a0,&3e,&1b,&cd,&a6,&a0,&3e,&4c
1420 DATA &cd,&a6,&a0,&3e,&7f,&cd,&a6,&a0
1430 DATA &3e,&02,&cd,&a6,&a0,&c9,&cd,&2e
1440 DATA &bd,&38,&fb,&cd,&2b,&bd,&c9,&3a
1450 DATA &c0,&a0,&fe,&07,&c8,&af,&cb,&11
1460 DATA &cb,&11,&cb,&11,&c9,&00,&00,&00
1470 '
1480 IF Z<>23151 THEN PRINT "ERREUR DANS LES DATAs":END
1490 '
1500 'SORTIR L'EN-TETE DU PROGRAMME
```

```
1510 '
1520 GOSUB 3460
1530 '
1540 'ENTRER LES COMMENTAIRES
1550 '
1560 LOCATE 36,8
1570 PRINT "|--------------------------------------|"
1580 LOCATE 1,8
1590 INPUT "TITRE (40 LETTRES MAXI)";UE$
1600 '
1610 LOCATE 36,10
1620 PRINT "|--------|"
1630 LOCATE 1,10
1640 INPUT "ECHELLE DE VALEURS (10 LETTRES MAXI)";WE$
1650 '
1660 'ENTREE DES DIFFERENTES VALEURS ET CALCUL DU MAXIMUM
1670 '
1680 LOCATE 1,12
1690 PRINT "ENTREZ MAINTENANT LES VALEURS POUR LES DIFFERENTS
MOIS!"
1700 PRINT
1710 '
1720 FOR I=1 TO 12
1730 READ MOIS$
1740 PRINT USING "\          \";MOIS$;
1750 INPUT MY(I)
1760 IF MA<MY(I) THEN MA=MY(I)
1770 NEXT I
1780 '
1790 CLS
1800 '
1810 'SORTIR L'EN-TETE DE PROGRAMME
1820 '
1830 GOSUB 3460
1840 '
1850 LOCATE 1,8
1860 PRINT "CHOISISSEZ UNE FORME DE REPRESENTATION:"
1870 PRINT
1880 PRINT "POINTS        (P)"
1890 PRINT "COURBE        (C)"
1900 PRINT "HISTOGRAMMES (H)"
1910 LOCATE 1,14
```

```
1920 PRINT "VOTRE CHOIX"
1930 '
1940 DA$=INKEY$
1950 IF DA$<>"P" AND DA$<>"C" AND DA$<>"H" AND DA$<>"p" AND DA$<>"c
" AND DA$<>"h" THEN 1910
1960 '
1970 CLS
1980 '
1990 'CALCULER LA POSITION DES COMMENTAIRES
2000 '
2010 T1=LEN(UE$)
2020 T2=LEN(WE$)
2030 PU=40-(T1/2)
2040 PW=6-(T2/2)
2050 '
2060 'SORTIR LES COMMENTAIRES
2070 '
2080 LOCATE 69,20
2090 PRINT "MOIS"
2100 LOCATE PU,1
2110 PRINT UE$
2120 LOCATE PW,9
2130 PRINT WE$
2140 '
2150 'DESSINER LE SYSTEME DE COORDONNEES
2160 '
2170 MOVE 163,375
2180 DRAW 163,112
2190 DRAW 530,112
2200 '
2210 'GRADUATION DE L'AXE DES X
2220 '
2230 FOR X=163 TO 515 STEP 32
2240 MOVE X,100
2250 DRAW X,375
2260 NEXT X
2270 '
2280 'COMMENTAIRE DE L'AXE DES X
2290 '
2300 FOR MY=1 TO 3
2310 MY1=MY+19
2320 RESTORE 2930
```

```
2330 '
2340 FOR M=1 TO 12
2350 MX=M*4+17
2360 READ MOIS$
2370 MOIS$=RIGHT$(LEFT$(MOIS$,MY),1)
2380 LOCATE MX,MY1
2390 PRINT MOIS$
2400 NEXT M
2410 '
2420 NEXT MY
2430 '
2440 'GRADUATION DE L'AXE DES Y
2450 '
2460 FOR Y=362 TO 112 STEP -25
2470 MOVE 155,Y
2480 DRAW 530,Y
2490 NEXT Y
2500 '
2510 'COMMENTAIRE DE L'AXE DES Y
2520 '
2530 IF INT(MA)<>MA THEN MA=INT(MA)+1
2540 IF INT(MA/2)<>MA/2 THEN MA=MA+1
2550 '
2560 TAG
2570 '
2580 FOR I=0 TO 10
2590 PY=25*I+117
2600 MOVE 95,PY
2610 PRINT USING "#####.#";MA/10*I;
2620 NEXT I
2630 '
2640 'FORME DE REPRESENTATION
2650 '
2660 IF DA$="P" OR DA$="p" THEN GOSUB 2950
2670 IF DA$="C" OR DA$="c" THEN GOSUB 3110
2680 IF DA$="H" OR DA$="h" THEN GOSUB 3290
2690 '
2700 TAGOFF
2710 '
2720 LOCATE 1,20
2730 PRINT "IMRIMER (I)"
2740 PRINT "RETOUR  (R)"
```

```
2750 PRINT "FIN      (F)"
2760 '
2770 HZE$=INKEY$
2780 IF HZE$<>"I" AND HZE$<>"R" AND HZE$<>"F" AND HZE$<>"i" AND HZE
$<>"r" AND HZE$<>"f" THEN 2720
2790 IF HZE$="F" OR HZE$="f" THEN CLS:END
2800 IF HZE$="R" OR HZE$="r" THEN CLS:GOTO 1790
2810 '
2820 'REALISER LA COPIE SUR IMPRIMANTE
2830 '
2840 LOCATE 1,20
2850 PRINT "                "
2860 PRINT "                "
2870 PRINT "                "
2880 '
2890 CALL &A000
2891 '
2892 GOTO 1790
2900 '
2910 'DATAs AVEC LES NOMS DE MOIS
2920 '
2930 DATA JANVIER,FEVRIER,MARS,AVRIL,MAI,JUIN
2940 DATA JUILLET,AOUT,SEPTEMBRE,OCTOBRE,NOVEMBRE,DECEMBRE
2950 '
2960 'FIXER LES POINTS
2970 '
2980 SYMBOL 255,0,65,34,20,8,20,34,65
2990 '
3000 F=131
3010 '
3020 FOR I=1 TO 12
3030 MY=((MY(I)/MA)*250)+112
3040 F=F+32
3050 MY=MY+8
3060 MOVE F-4,MY
3070 PRINT CHR$(255);
3080 NEXT I
3090 '
3100 RETURN
3110 '
3120 'DESSINER LA COURBE
3130 '
```

```
3140 F=131
3150 '
3160 FOR I=1 TO 12
3170 MY=((MY(I)/MA)*250)+112
3180 F=F+32
3190 IF I=1 THEN X2=F:Y2=MY
3200 X1=X2
3210 Y1=Y2
3220 X2=F
3230 Y2=MY
3240 MOVE X1,Y1
3250 DRAW X2,Y2
3260 NEXT I
3270 '
3280 RETURN
3290 '
3300 'DESSINER LES HISTOGRAMMES
3310 '
3320 F=131
3330 '
3340 FOR I=1 TO 12
3350 MY=((MY(I)/MA)*250)+112
3360 F=F+32
3370 '
3380 FOR L=F-7 TO F+7
3390 MOVE L,MY
3400 DRAW L,112
3410 NEXT L
3420 '
3430 NEXT I
3440 '
3450 RETURN
3460 '
3470 'SORTIR L'EN-TETE DE PROGRAMME
3480 '
3490 LOCATE 30,3
3500 PRINT "GRAPHIQUE STATISTIQUE 2 DIM."
3510 LOCATE 33,4
3520 PRINT "JST  30.7.1986"
3530 '
3540 'DESSINER LE CADRE
3550 '
```

```
3560 MOVE 16,383
3570 DRAW 16,320
3580 DRAW 623,320
3590 DRAW 623,383
3600 DRAW 16,383
3610 '
3620 RETURN
```

Description du programme :

1000-1130 Définitions, dimensionnements et pré-affectations. L'écran est placé en mode 80 colonnes et la zone de la RAM disponible pour le BASIC est limitée avec MEMORY &8000. Au-delà de cette limite, nous pourrons placer un programme machine qui sera ainsi protégé contre tout empiètement du BASIC. Nous choisissons une combinaison de couleurs riche en contraste et la variable MY est dimensionnée à 13 éléments. Les valeurs pour les douze mois seront stockées sous les indices 1 à 12. La variable MA qui devra recevoir au cours du programme le maximum des douze valeurs mensuelles est fixée sur zéro.

1140-1480 *Copie d'écran graphique*
Cette partie du programme écrit dans la mémoire une routine simple de copie d'écran qui permettra de sortir sur imprimante les graphiques réalisés. La routine figure dans la mémoire RAM à partir de l'adresse &A000 et elle peut être appelée avec CALL &A000. Le chapitre 8.3 décrit cette routine plus précisément.

1490-1520 *Sortir l'en-tête du programme*
Le titre du programme est sorti sur l'écran et encadré. Le sous-programme en ligne 3460 est appelé à cet effet.

1530-1640 *Entrer les commentaires*
Le titre et les commentaires de l'échelle des valeurs du graphique à réaliser sont entrés avec l'instruction INPUT. La place réservée aux entrées est marquée sur l'écran.

1650-1790 *Entrée des différentes valeurs et calcul du maximum*
Après un message indiquant qu'il s'agit maintenant d'entrer les valeurs pour les différents mois, le programme commence le traitement d'une boucle FOR-TO-NEXT. Dans cette boucle, qui est parcourue exactement 12 fois, on demande à l'utilisateur d'effectuer les entrées correspondant aux différents mois. A cet effet, la variable MOIS$ se voit attribuer lors de chaque parcours de la boucle un des noms de mois placés dans le bloc de DATA à la fin du programme.

Après que le nom du mois ait été sorti sous un format uniforme pour tous les mois, le programme attend la valeur qui servira de base à la représentation graphique du mois correspondant. La ligne 1760 compare le contenu actuel de MY(I) à celui de MA. Si MA contenait la valeur la plus petite, elle se voit affecter la valeur la plus grande des deux. Après que la boucle FOR-TO-NEXT ait été parcourue entièrement, MA contiendra donc la valeur maximale des douze mois.

Après que toutes les indications nécessaires aient été collectées, l'écran est vidé pour permettre les sorties de messages suivantes.

1800-1970 Après une nouvelle sortie de l'en-tête du programme (GOSUB 3460) cette partie du programme présente sur l'écran une sélection de trois points du menu. En actionnant les touches "P", "C" ou "H", l'utilisateur choisit entre les formes de représentation points, courbes ou histogrammes. Cette sélection effectuée, l'écran est vidé pour le dessin du graphique.

1980-2040 *Calculer la position pour le titre*
La longueur du titre et du commentaire de l'échelle de valeurs sont déterminées à l'aide de la fonction LEN et elles sont affectées aux variables T1 et T2. Le contenu de ces variables divisé par 2 est chaque fois soustrait de la position X qui devra coïncider par la suite avec le milieu de l'inscription. On peut ainsi en déduire la position X du curseur de texte à laquelle devra commencer la sortie des textes pour

qu'ils apparaissent centrés dans l'emplacement voulu. Les coordonnées X pour le début des sorties de texte se trouvent dans les variables PU (titre) et PW (échelle de valeurs).

2050-2130 *Sortir le commentaire*
Le commentaire pour le graphique est sorti sur l'écran en utilisant les positions de sortie de texte calculées.

2140-2190 *Dessiner le système de coordonnées*
Le système de coordonnées sur lequel reposera le graphique est dessiné avec une instruction MOVE et deux instructions DRAW.

2200-2260 *Graduation de l'axe des X*
Après que cette partie du programme ait été exécutée, le système de coordonnées se trouve divisé par 12 lignes verticales partant de l'axe des X. Cette subdivision fondée sur les 12 mois est effectuée pour rendre le graphique plus lisible.

2270-2420 *Commentaires de l'axe des X*
Cette partie du programme écrit les trois premières lettres des noms de mois l'une sous l'autre verticalement dans les emplacements prévus à cet effet sur l'axe des X. Ce travail est effectué par deux boucles FOR-TO-NEXT imbriquées.

La boucle extérieure est parcourue exactement trois fois et elle traite chaque fois une des trois premières lettres des noms de mois. La variable de comptage de cette boucle (MY) permet en ligne 2310 de déterminer la position Y pour la sortie des lettres correspondant au parcours actuel de la boucle. Le pointeur DATA est réinitialisé avant l'entrée dans la boucle FOR-TO-NEXT intérieure. Cela permet de relire les noms de mois lors de chaque parcours de boucle.

Chacun des 12 parcours de la boucle intérieure traite un mois déterminé. La boucle calcule d'abord d'après la variable de comptage M la coordonnée X pour la

sortie de la lettre. Cette coordonnée est affectée à MX. La position de la sortie de texte figure ainsi dans les variables MX et MY1. Les lignes 2360 et 2370 déterminent quelle lettre doit être sortie.

La ligne de programme 2370 isole la lettre voulue du nom de mois tiré du bloc de DATA. On détache pour cela, en fonction de la variable de comptage MY, la première lettre, les deux premières lettres ou les trois premières lettres du nom du mois (LEFT$(MOIS$,MY)). On réserve chaque fois la dernière de ces lettres pour la sortie. Le curseur de texte est alors positionné dans l'emplacement calculé et la lettre isolée qui se trouve dans la variable MOIS$ est sortie.

2430-2490 *Graduation de l'axe des Y*
Cette partie du programme divise le système de coordonnées par 11 lignes horizontales partant de l'axe des Y. La ligne la plus basse (l'axe des X) représente le point zéro sur l'échelle des valeurs. Toutes les autres divisions servent à obtenir une attribution plus fine des indications de l'échelle aux points du graphique. C'est pourquoi elles coupent tout le graphique comme les divisions verticales.

2500-2620 *Commentaires sur l'axe des Y*
Le premier travail de cette partie du programme est de déterminer la valeur maximale (MA) résultant des valeurs entrées pour les différents mois. La ligne 2530 élimine les décimales éventuelles et augmente la valeur maximale de un si ce n'est pas une valeur entière. Cela est indispensable pour qu'on ne puisse pas aboutir à des valeurs supérieures à la graduation la plus élevée de l'échelle. Si la valeur maximale est un nombre impair, elle est augmentée de un en ligne 2540 pour éviter une subdivision trop complexe de l'échelle des valeurs.

Après que la sortie de texte ait été reliée au curseur graphique (TAG), une boucle FOR-TO-NEXT écrit les commentaires sur l'axe des Y. Lors de chacun des 11

parcours de la boucle, on calcule d'abord dans quelle position le texte pour le parcours actuel devra être sorti. Cette position est affectée à la variable PY. L'inscription se fait de bas en haut. C'est chaque fois la valeur de la variable de comptage multipliée par le dixième de la valeur maximale qui est sortie.

2630-2680 *Forme de représentation*
Cette partie du programme saute aux sous-programmes de dessin de points (ligne 2950), de courbe (ligne 3110) ou d'histogrammes (ligne 3290) en fonction de l'entrée effectuée en réponse à la question sur la forme de représentation.

2690-2800 Une fois que le graphique voulu se trouve sur l'écran, un menu apparaît qui vous offre la possibilité d'imprimer une copie du graphique qui figure sur l'écran ou bien de revenir à la sélection d'un mode de représentation ou encore de sortir du programme.

Après que vous ayez sélectionné l'un de ces trois points, le programme se termine (ligne 2790) ou bien il revient à la ligne 1790 ou bien enfin, si aucun des deux points précédents n'a été sélectionné, la suite du programme, c'est-à-dire la copie d'écran, est exécutée.

2810-2890 *Réaliser une copie d'écran*
Pour débarrasser le graphique de toute mention inutile, les lignes 2840 à 2870 effacent le menu sur l'écran et appellent ensuite la routine de copie d'écran (CALL &A000). Après impression de la copie d'écran, le programme revient à la ligne 1790.

2900-2940 *DATAs avec les noms de mois*
C'est ici que figurent, sur deux lignes de DATA, les noms des douze mois qui sont utilisés par les parties ENTRER LES DIFFERENTES VALEURS ET CALCULER LA VALEUR MAXIMALE et COMMENTAIRE DE L'AXE DES X.

2950-3100 *Fixer les points*

Après que le caractère 255 ait été redéfini pour recevoir la forme d'une croix (ligne 2980), une boucle FOR-TO-NEXT dessine les croix qui constituent le graphique proprement dit. La ligne 3030 calcule d'abord la position Y pour le centre de la croix à partir de la valeur correspondant au mois désigné par la variable de comptage. Cette valeur pour le mois est calculée à partir de la valeur maximale et de deux constantes.

La position X de la croix est obtenue en ajoutant 32, lors de chaque parcours de la boucle, à la variable F qui a été fixée sur la valeur défaut 131 avant la boucle. Les deux coordonnées calculées doivent être converties de façon à obtenir les coordonnées du coin supérieur gauche de la matrice de caractère puisqu'il ne s'agit pas de sortir un point mais bien un caractère.

Cela est fait en ligne 3050 pour la position Y. La position X est adaptée lors du déplacement du curseur graphique (ligne 3060) car elle ne doit pas être réécrite. La croix définie sous le code de caractère 255 est placée sur l'écran dans la position du curseur graphique ainsi obtenue.

La valeur correspondant à chaque mois est marquée par l'intersection des deux lignes de la croix.

Après que le graphique ait été réalisé, retour du sous-programme (ligne 3100).

3110-3280 *Dessin d'une courbe*

La structure du sous-programme DESSIN DE LA COURBE corrrespond dans ses grandes lignes à la structure de FIXER LES POINTS. La principale différence est que les points calculés seront reliés entre eux par des traits. A cet effet, les coordonnées du point précédent sont prises comme coordonnées de départ pour le tracé menant au point actuel.

Lors du premier parcours de la boucle FOR-TO-NEXT, il ne peut y avoir encore de coordonnées de départ de sorte que celles-ci doivent être produites artificiellement (ligne 3190).

3290-3450 *Dessin des histogrammes*
DESSIN DES HISTOGRAMMES a également une structure semblable à celle de FIXER LES POINTS, si ce n'est qu'on ne tracera pas ici des croix mais des histogrammes allant de l'axe des X aux valeurs calculées pour les différents mois. On intègre à cet effet une seconde boucle FOR-TO-NEXT dans ce sous-programme, boucle qui se chargera de dessiner les douze histogrammes.

La valeur de départ pour la seconde boucle est obtenue en diminuant de sept la position X dans laquelle figure le mois actuellement traité et la valeur finale est obtenue en augmentant cette même position de sept. Le curseur graphique est placé, lors de chaque parcours de la boucle, sur la position définie par la variable de comptage de la boucle (L) et la valeur Y calculée pour le mois.

Une ligne est alors tracée de cette position à l'axe des X. Une fois que la boucle FOR-TO-NEXT intérieure a terminé son travail, l'histogramme pour le mois indiqué par la boucle extérieure (I) a été dessiné.

3460-Fin *Sortir l'en-tête du programme*
Ce sous-programme sort et encadre sur l'écran le titre du programme. Ensuite retour au programme principal.

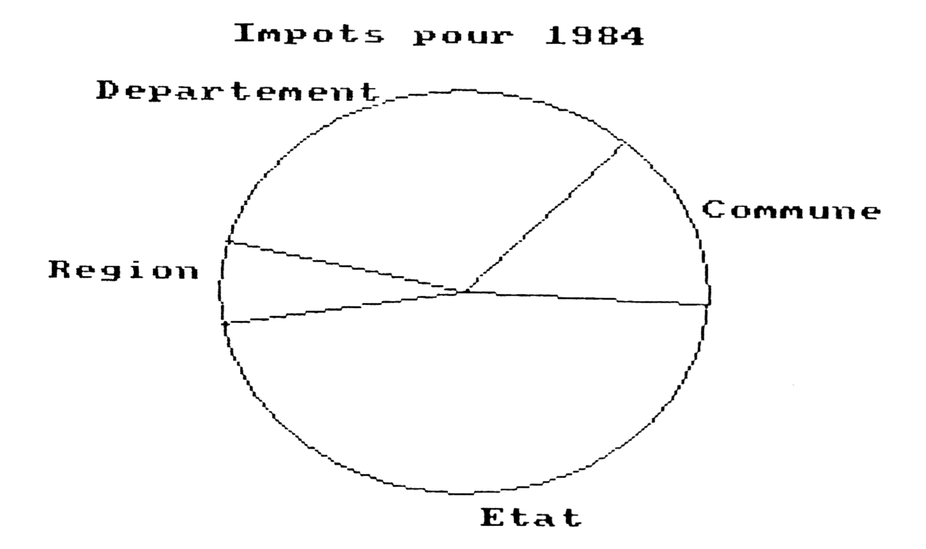
5.3 GRAPHIQUE CAMEMBERT

Les graphiques camembert constituent une forme d'illustration des statistiques qui se différencient des méthodes décrites jusqu'ici. Le graphique camembert permet de représenter le rapport en pourcentage entre différents nombres qui peuvent être regroupés pour former une entité.

Les pourcentages sont représentés ici par des sections d'un cercle complet de 360 degrés, d'où le nom de graphiques "camembert".

Pour vous montrer comment des statistiques peuvent être illustrées par un graphique camembert, nous prendrons un exemple tiré du budget de l'Allemagne de l'Ouest :

En 1984, la fédération, les régions, les communes et la communauté européenne perçurent 415 milliards de Deutsch Marks. 48% de cette somme (199,2 milliards) revinrent à la fédération, 34,7% (144,005 milliards) aux régions, 13% (53,95 milliards) aux communes et 3,5% (14,525 milliards) à la communauté européenne (d'après : Kaufmännische Betriebslehre (H), Europ Lehrmittel, 1986, page 493). Pour réaliser le graphique, on considère que les 415 milliards représentent 100%, ce qui correspondra sur le graphique au cercle complet de 360 degrés. On affectera ensuite aux différents destinataires des impôts perçus en RFA une part de camembert proportionnelle au pourcentage qui leur revient. C'est ainsi qu'on aboutit au graphique suivant :



<u>Copie d'écran 11 :</u> **Le programme "graphique camembert"**

Le programme graphique "camembert", avec lequel ce graphique a été réalisé, attend de l'utilisateur d'abord l'entrée d'un titre, puis l'entrée des valeurs à représenter ainsi que leurs désignations. Peu importe pour le programme que les nombres entrés soient des valeurs absolues ou déjà des pourcentages car le programme calculera de toute façon la proportion du cercle complet revenant à chacune de ces grandeurs. Si l'on veut mettre fin à l'entrée des valeurs, il suffit de répondre en appuyant simplement sur la touche RETURN lorsque le programme vous demande d'entrer une valeur et sa désignation.

Une fois que toutes les valeurs ont été correctement entrées, à condition qu'il y en ait au moins deux, le programme commence à dessiner le graphique. Il sort d'abord le titre et dessine le cercle complet. Le cercle est ensuite divisé en différentes sections dont chacune reçoit un intitulé. Une fois le graphique terminé, il reste à l'écran jusqu'à ce que le programme soit interrompu avec la touche ESC.

```
100 MODE 2
110 DEG
120 DIM VALEUR(25)
130 DIM DEGRE(25)
140 DIM P(25)
150 DIM NOM$(25)
160 WINDOW#1,1,80,10,25
170 GES=0
180 INDEX=1
190 '
200 'SORTIR LE TITRE DU PROGRAMME
210 '
220 LOCATE 32,3
230 PRINT "GRAPHIQUE CAMEMBERT"
240 LOCATE 34,4
250 PRINT "JST 2.8.1986"
260 '
270 'DESSIN DU CADRE
280 '
290 MOVE 16,383
300 DRAW 16,320
310 DRAW 623,320
320 DRAW 623,383
```

```
330 DRAW 16,383
350 '
360 'ENTREE DES COMMENTAIRES
370 '
380 LOCATE#1,26,1
390 PRINT#1,"|------------------------------------|"
400 LOCATE#1,1,1
410 INPUT#1,"TITRE (38 LETTRES MAXI)";UE$
420 T1=LEN(UE$)
430 PU=20-(T1/2)
440 '
450 CLS#1
460 '
470 LOCATE 1,8
480 PRINT "ENTRER MAINTENANT LES VALEURS ET LES NOMS!"
490 '
500 PRINT#1," VALEUR ";USING"##";INDEX;
510 PRINT#1,":";
520 INPUT#1,VALEUR(INDEX)
530 '
540 PRINT#1,"NOM:";
550 INPUT#1,NOM$(INDEX)
560 '
570 IF VALEUR(INDEX)=0 AND NOM$(INDEX)="" THEN CLS:GOTO 650
580 '
590 GES=GES+VALEUR(INDEX)
600 INDEX=INDEX+1
610 PRINT#1
620 '
630 GOTO 500
640 '
650 IF INDEX<=2 THEN END
660 '
670 MODE 1
680 '
690 'SORTIR LES COMMENTAIRES
700 '
710 LOCATE PU,1
720 PRINT UE$
730 '
740 'DESSINER LE GRAPHIQUE
750 '
```

```
760 GOSUB 1110
770 '
780 ORIGIN 320,200
790 TAG
800 '
810 FOR I=1 TO INDEX
820 DEGRE(I)=INT(VALEUR(I)*360/GES)
830 P(I)=INT(DEGRE(I)/2)
840 NEXT I
850 '
860 I=1:ALT=0
870 FOR DEGRE=1 TO 360
880 MOVE 0,0
890 IF DEGRE=ALT+P(I) THEN ALT=ALT+P(I)*2:GOSUB 930
900 IF DEGRE(I)=DEGRE THEN DRAW 150*COS(DEGRE),150*SIN(DEGRE):DEGRE
(I+1)=DEGRE(I+1)+DEGRE(I):I=I+1
910 NEXT DEGRE
920 GOTO 920
930 '
940 'COMMENTAIRES POUR LES SECTIONS DE CERCLE
950 '
960 O=(LEN(NOM$(I))*16)+10
970 X=150*COS(DEGRE)
980 Y=150*SIN(DEGRE)
990 MOVE X,Y
1000 '
1010 IF X>=0 THEN MOVER 10,0
1020 IF X<0  THEN MOVER -O,0
1030 IF Y>=0 THEN MOVER 0,10
1040 IF Y<0  THEN MOVER 0,-10
1050 '
1060 PRINT NOM$(I);
1070 '
1080 MOVE 0,0
1090 '
1100 RETURN
1110 '
1120 'DESSINER LE CERCLE
1130 '
1140 'CENTRE ET RAYON
1150 '
1160 XP%=320
```

```
1170 YP%=200
1180 XR%=150
1190 YR%=150
1200 '
1210 'PREMIERE COORDONNEE
1220 '
1230 j%=0
1240 GOSUB 1490
1250 x1%=x2%
1260 y1%=y2%
1270 '
1280 'AUTRES COORDONNEES
1290 '
1300 FOR j%=0 TO 90 STEP 6
1310 GOSUB 1490
1320 '
1330 'DESSIN DU CERCLE
1340 '
1350 MOVE xp%+x1%,yp%+y1%
1360 DRAW xp%+x2%,yp%+y2%
1370 MOVE xp%+x1%,yp%-y1%+1
1380 DRAW xp%+x2%,yp%-y2%+1
1390 MOVE xp%-x1%+1,yp%-y1%+1
1400 DRAW xp%-x2%+1,yp%-y2%+1
1410 MOVE xp%-x1%+1,yp%+y1%
1420 DRAW xp%-x2%+1,yp%+y2%
1430 '
1440 x1%=x2%:y1%=y2%
1450 '
1460 NEXT
1470 '
1480 RETURN
1490 '
1500 'CALCUL DES COORDONNEES
1510 '
1520 x2%=INT(COS(j%)*xr%+0.5)
1530 y2%=INT(SIN(j%)*yr%+0.5)
1540 '
1550 RETURN
```

Description du programme :

100-180 Définitions, dimensionnements et initialisation de variables. Le MODE 2 est sélectionné et l'ordinateur est fixé avec DEG sur la mesure en degrés. Les variables VALEUR, DEGRE, P et NOM$ sont dimensionnées chacune à 26 éléments ce qui correspond au nombre maximal de parts de camembert pouvant être représentées avec notre programme.

Une fenêtre est défini sous #1. C'est à l'intérieur de cette fenêtre que les valeurs à représenter et leurs désignations pourront être entrées sans endommager le titre.

Les variables GES et INDEX sont fixées sur leurs valeurs par défaut. GES servira au calcul de la somme des valeurs à représenter et INDEX constituera la clé d'accès aux différentes variables de tableau.

190-250 *Sortir le titre du programme*
Le titre du programme est sorti sur l'écran.

260-330 *Dessin du cadre*
Les sorties sur l'écran sont encadrées d'une ligne.

350-670 *Entrée des commentaires*
Une zone de 38 caractères est marquée sur l'écran pour que l'utilisateur y entre le texte du titre du graphique camembert. Une fois le texte entré, sa longueur est calculée en ligne 420. D'après cette longueur, le programme calcule la position X dans laquelle devra commencer la sortie du texte sur l'écran pour que le texte apparaisse centré. Toutes les entrées ont été effectuées jusqu'ici dans la fenêtre #1 qui est maintenant effacée en ligne 450.

Le message "ENTREZ MAINTENANT LES VALEURS ET LES NOMS!" invite l'utilisateur à entrer les données à représenter. Ce message devra rester visible sur l'écran pendant toute la phase d'entrée et c'est pourquoi il est sorti dans la fenêtre #0.

Les valeurs et leurs désignations sont ensuite entrées dans la fenêtre #1. Lorsque la dernière ligne de cette fenêtre est pleine, on peut faire glisser l'affichage à l'intérieur de #1 sans détruire le titre.

La ligne 570 teste si la valeur entrée est 0 et si le nom correspondant est une chaîne vide. Si c'est le cas, on sort de la boucle d'entrée.

Si la condition d'interruption n'est pas remplie, le programme additionne la valeur entrée aux valeurs entrées précédemment dont la somme est dans la variable GES. Après que l'index ait été incrémenté (INDEX), le traitement de la boucle d'entrée recommence au début.

Après avoir quitté la boucle d'entrée, le programme saute à la ligne 650 où il rencontre un END si moins de deux valeurs ont été entrées. Vous comprenez bien sûr qu'un graphique camembert n'a aucun intérêt pour moins de deux valeurs!

680-720 *Sortir les commentaires*
Les commentaires du graphique camembert sont sortis sur l'écran, qui a été vidé auparavant par un MODE 1 (ligne 670), dans l'emplacement calculé précédemment.

730-920 *Dessin du graphique*
La première étape de la réalisation d'un graphique camembert consiste à appeler le sous-programme en ligne 1110 qui dessine un cercle sur l'écran. L'origine des coordonnées est alors fixée sur le centre de ce cercle et la sortie de texte est reliée au curseur graphique.

Une boucle FOR-TO-NEXT, qui sera parcourue autant de fois qu'il y a de valeurs entrées, calcule combien de degrés du cercle devront être attribués à la représentation des différentes valeurs. On calcule ensuite la moitié de ces indications de degrés pour définir la position pour les titres des différentes sections de cercle.

La boucle FOR-TO-NEXT de la ligne 870 à la ligne 910 se charge du découpage du cercle et de l'écriture des différents titres de sections. La boucle est parcourue exactement 360 fois ce qui permet d'examiner pour chaque degré du cercle s'il faut inscrire un titre ou délimiter une section de cercle. Les variables I et ALT, qui ont été fixées sur leurs valeurs par défaut avant l'entrée dans la boucle, sont utilisées comme variables auxiliaires. I est un indice qui permet d'accéder aux différents éléments des variables indicées P et DEGRE. La variable ALT contient toujours l'indication en degrés de la limite extérieure de la dernière section de cercle à avoir été écrite.

Une fois que le graphique camembert a été réalisé sur l'écran, la ligne 920 fait entrer l'ordinateur dans une boucle sans fin pour que le message "Ready" ne puisse pas détruire le graphique.

930-1100 *Ecriture des commentaires de sections de cercle*
Ce sous-programme est appelé lors du dessin du graphique camembert chaque fois qu'on atteint la moitié d'une section de cercle. Le sous-programme calcule la position dans laquelle le titre de la section de cercle devra être sorti et sort le texte qui est contenu dans NOM$ en fonction de l'indice actuel. Ce travail terminé, la sous-routine replace le curseur graphique sur l'origine des coordonnées avant retour du sous-programme.

1110-Fin *Dessin du cercle*
Les variables entières XP% (coordonnée X du centre) et YP% (coordonnée Y du centre), XR% (rayon dans le sens des X) et YR% (rayon dans le sens des Y) reçoivent les valeurs appropriées pour le dessin du cercle.

Comme le dessin du cercle ne se fait pas avec l'instruction BASIC PLOT mais avec l'instruction DRAW, il faut calculer chaque fois un point de départ et un point final pour la ligne. Le point de départ de la première ligne (0 degré) est calculé dans la partie PREMIERE COORDONNEE du programme.

Les coordonnées se retrouveront finalement dans les variables X1% et Y1%. La boucle FOR-TO-NEXT des lignes 1300 à 1460 calcule ensuite toutes les autres coordonnées (de 6 à 90 degrés) par pas de 6 degrés.

Notons encore, en ce qui concerne les coordonnées, qu'elles se trouvent toutes sans exception dans le premier quart de cercle. Les coordonnées pour les trois autres quarts de cercle sont calculées par symétrie par rapport au centre du cercle. Si vous voulez savoir précisément comment se fait le dessin du cercle, vous le comprendrez vite en examinant la partie DESSINER CERCLE du programme. Une fois qu'une ligne a été tracée, son point extrême est réutilisé comme point de départ pour la prochaine ligne (ligne 1440). Après que le cercle ait été entièrement dessiné, retour à la boucle principale.

Le sous-programme CALCUL DES COORDONNEES est utilisé exclusivement par DESSINER CERCLE. Ce sous-programme permet de calculer les coordonnées des points situés sur la circonférence du premier quart de cercle grâce aux outils de la géométrie.

J% contient une indication de degrés par rapport à laquelle sont calculées les coordonnées (X2%, Y2%) correspondantes.

6. Représentation graphique de fonctions mathématiques

Vous souvenez-vous encore du temps où vous alliez à l'école? si oui, vous vous souvenez certainement d'un épisode de cours de maths qu'on pourrait intituler ainsi : "qui dessine la plus belle courbe de fonction?". Si vous avez déjà oublié cette époque, alors laissez-vous aller et essayons ensemble de regrouper nos souvenirs.

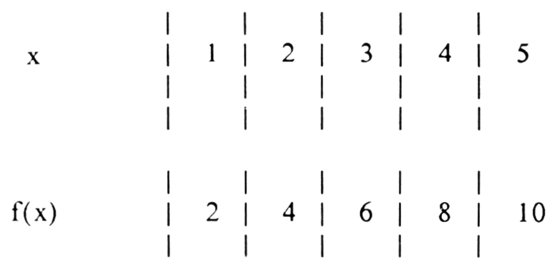
La représentation graphique de fonctions mathématiques est en général autant appréciée des professeurs que des élèves. Les élèves trouvent en effet qu'il est plus agréable de dessiner que de faire des calculs et les professeurs aiment bien les fonctions dessinées parce que cela leur permet enfin de donner à leurs élèves des notes en fonction de leur propreté. Mais il est temps pour nous maintenant de nous dégager de toutes impressions émotionnelles et de commencer ce chapitre en expliquant d'abord ce qu'est une fonction.

Une fonction peut être définie comme une relation univoque entre des éléments de deux ensembles A et B. On peut naturellement utiliser n'importe quelle lettre pour désigner une fonction; on utilise cependant souvent la lettre f, parfois aussi g, h, etc. si une fonction f affecte à un élément x de A l'élément y de B, on pourra dire que y est la valeur de fonction de x. y pourra de ce fait être désigné par f(x), qui doit être lu : f de x. La variable x est aussi appelée l'argument de la fonction.

Si cette définition ne vous a pas paru limpide, nous allons essayer de vous décrire ce principe une nouvelle fois, de façon plus concrète, en prenant un exemple. Prenons comme domaine de définition, c'est-à-dire pour l'ensemble A de la définition donnée ci-dessus, l'ensemble de nombres {1;2;3;4;5}. On affectera maintenant des éléments du domaine de valeurs B à cet ensemble de nombres, à travers l'équation de fonction f(x)=2*x, B étant défini comme l'ensemble des nombres entiers. L'ensemble des nombres entiers est l'ensemble qui se compose des nombres 1, 2, 3, ... jusqu'à l'infini.

Une table de valeurs nous permettra de représenter de façon claire les relations qui résultent de l'équation de la fonction. La table

de valeurs peut être réalisée tout simplement en remplaçant x dans l'équation de la fonction par tous les éléments du domaine de définition l'un après l'autre et en calculant chaque fois le résultat de la fonction f(x). On établit alors une liste des valeurs respectives de x en les mettant en relation avec les valeurs correspondantes de f(x). La table de valeurs est alors prête.



x	1	2	3	4	5
f(x)	2	4	6	8	10

Pour revenir à la question *"Qui dessine la plus belle courbe de fonction?"* que nous évoquions en introduction, précisons maintenant que la relation mathématique présentée par la table de valeurs peut aussi être représentée graphiquement. Pour ne pas nous égarer dans des considérations d'ordre théorique, restons-en à notre exemple f(x)=2*x.

Pour représenter cette relation sous forme d'un dessin, il suffit au fond de disposer de deux droites se coupant en angle droit. Les deux droites représenteront des flux de nombres, la droite horizontale représentant les éléments de l'ensemble de définition et la droite verticale ceux du domaine de valeurs.

Les deux flux de nombres seront gradués à intervalles réguliers en partant de leur point d'intersection. Cette graduation sera marquée de la façon suivante : la droite horizontale, que nous appellerons aussi l'axe des X, sera marquée 1, 2, 3, etc. de gauche à droite, à partir du point d'intersection qui sera marqué zéro. Nous procéderons de même avec la seconde ligne mais ici de bas en haut. Le dessin ainsi réalisé est appelé en mathématiques "le système de coordonnées cartésien".

C'est dans ce système de coordonnées que nous allons maintenant placer le graphe de notre fonction, c'est-à-dire un dessin la représentant. Prenons le premier élément du domaine de définition, le 1. En partant de l'axe des X, c'est-à-dire de l'axe contenant

les éléments du domaine de définition, nous allons imaginer une droite verticale qui partirait du point marqué 1 sur l'axe des X. La table de valeurs nous indique que c'est l'élément 2 du domaine de valeurs qui est attribué au 1 de l'ensemble de définition. Cherchons donc sur l'axe vertical l'emplacement marqué 2 et imaginons une droite horizontale partant de là. Nous marquerons maintenant le point situé à l'intersection des deux lignes imaginaires.

Si nous procédons de même pour les cinq éléments du domaine de définition, nous obtenons l'image montrée à la figure 8.

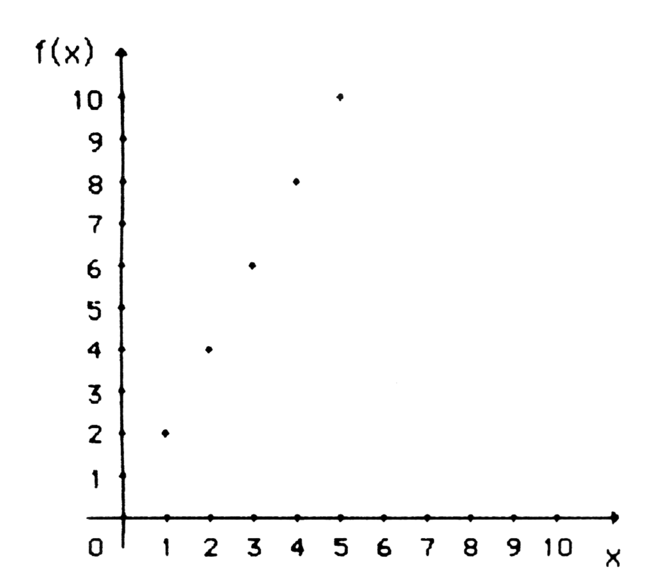


Figure 8 : **Graphe de la fonction f(x)=2*x (x tiré de {1;2;3;4;5})**

Les cinq points représentent sous une forme graphique la relation entre les éléments du domaine de définition et les éléments du domaine de valeurs. C'est pourquoi on parlera de graphe de fonction.

6.1 LE SYSTEME DE COORDONNEES

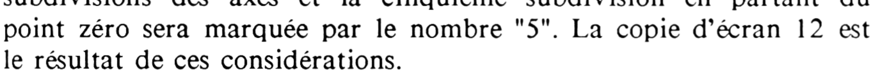
Les explications précédentes sur les fonctions et leur représentation nous ont appris que le système de coordonnées est la base sur laquelle repose la représentation graphique des fonctions mathématiques. C'est pourquoi la première étape du dessin d'un graphe de fonction sur l'ordinateur, qui constitue l'objet du présent chapitre, consistera à créer le système de coordonnées approprié.

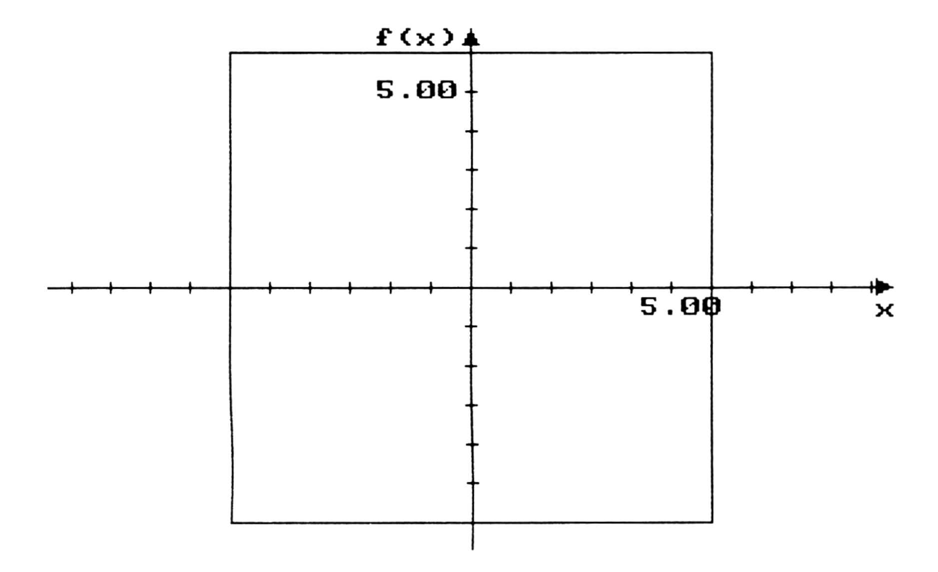
Essayons maintenant de préciser les conditions que doit remplir un système de coordonnées qui permette de dessiner le plus grand nombre de graphes de fonction possible. Il est bon pour cela d'observer la façon de procéder d'un homme lorsqu'il trace à la main un tel système de coordonnées.

Il n'est pas nécessaire de réfléchir avant de dessiner les deux axes qu'on retrouvera dans n'importe quel système de coordonnées. On place des flèches sur l'extrémité droite de l'axe des X et sur l'extrémité supérieure de l'axe des Y ou axe de f(x). Ces flèches indiquent que les lignes dessinées ne constituent que des portions des flux de nombres représentés mais que ceux-ci continuent à l'infini. Nous pouvons également donner un titre aux deux axes, à savoir "x" et "f(x)". L'étape suivante est la graduation des axes qui peut reposer sur l'unité de grandeur de votre choix mais qui doit être la même pour les deux axes. Jusqu'ici, le système de coordonnées sera identique pour toutes les fonctions.

Pour que la graduation des axes ne soit pas seulement constituée par une subdivision en petites unités, il faut donner un intitulé à cette graduation pour qu'elle fasse référence à un ordre de grandeur qui la rendra plus parlante. Il suffit pour cela de marquer sur certaines des subdivisions les éléments du domaine de définition (axe des X) ou du domaine de valeurs (axe des Y). Ce marquage ne peut être effectué de la façon la plus appropriée qu'une fois que l'on sait pour quel domaine de définition la fonction doit être dessinée.

On marquera généralement les graduations de telle façon que le domaine de définition voulu apparaisse aussi grand que possible dans le système de coordonnées. Si l'on veut examiner une fonction dans un domaine de définition allant de -6 à 6, on marquera 10


subdivisions des axes et la cinquième subdivision en partant du point zéro sera marquée par le nombre "5". La copie d'écran 12 est le résultat de ces considérations.



<u>Copie d'écran 12 :</u> **Système de coordonnées pour représenter une fonction dans le domaine de définition de -6 à 6.**

Le programme avec lequel a été réalisée la sortie sur écran ci-dessus dessine les axes, les gradue et les subdivise. Il ne réclame de l'utilisateur que deux indications avant de commencer son travail. La première indication est le domaine de définition dont nous avons déjà parlé dans nos réflexions théoriques. La seconde indication est un facteur d'agrandissement. Ce facteur d'agrandissement permettra d'insérer au mieux dans la fenêtre de l'écran le domaine de définition à étudier.

Pour réaliser la copie d'écran 12, nous avons indiqué 1 comme facteur d'agrandissement, ce qui correspond à une sortie à l'échelle de 1 sur 1. Cette indication d'échelle est la plus adaptée car le programme subdivise d'une manière générale l'axe des X en 10 unités à partir du point zéro. On pourra donc ainsi représenter le domaine de définition de -6 à 6.

Si l'on veut toutefois examiner un domaine de définition de -1 à 1, ce domaine n'occuperait qu'une petite partie de l'écran. Il conviendra donc dans ce cas d'indiquer un facteur d'agrandissement de 10. La représentation du domaine de définition voulu prendra alors 10 fois plus de place et l'écran sera ainsi pratiquement rempli.

On peut naturellement utiliser également un nombre inférieur à 1 comme facteur d'agrandissement mais dans ce cas, bien évidemment, la représentation sur l'écran sera au contraire plus ramassée. Cela permettra de visualiser sur l'écran un domaine de définition de par exemple -100 à 100.

Lorsque vous choisissez le facteur d'agrandissement approprié, vous pouvez vous aider du cadre dans lequel est dessiné le système de coordonnées (voir la copie d'écran 12). Ce cadre délimite le domaine de définition sélectionné sur l'axe des X et délimite une portion de l'axe des Y symétrique à ce domaine. Si le cadre a l'aspect d'un petit carré perdu au milieu de l'écran, c'est qu'il faut choisir un nombre plus élevé comme facteur d'agrandissement. Si le cadre n'est pas visible, le facteur doit être diminué.

```
100 MODE 1
102 SYMBOL 255,8,8,28,28,62,62,127,8
104 '
106 'QUEL DOMAINE FAUT-IL EXAMINER
108 '
110 INPUT "DOMAINE DE DEFINITION SOUHAITE";DEFI
112 INPUT "AGRANDISSEMENT SOUHAITE";FF
114 FF=FF*30
116 '
118 CLS
120 '
122 'DESSIN DU SYSTEME DE COORDONNEES
124 '
126 TAG
128 ORIGIN 320,200
130 '
132 'AXE DES X
134 '
136 PLOT -320,0,1
```

```
138 DRAW 320,0
140 MOVER -16,6
142 PRINT CHR$(246);
144 MOVER -16,-16
146 PRINT "x";
148 '
150 'GRADUATION DE L'AXE DES X
152 '
154 FOR I=-300 TO 300 STEP 30
156 MOVE I,4
158 DRAW I,-4
160 NEXT I
162 '
164 'SUBDIVISION DE L'AXE DES X
166 '
168 MOVE 78,-8
170 PRINT USING "####.##";150/FF;
172 '
174 'AXE DES Y
176 '
178 MOVE 0,-200
180 DRAW 0,200
182 MOVER -8,-2
184 PRINT CHR$(255);
186 MOVER -80,0
188 PRINT "f(x)";
190 '
192 'GRADUATION DE L'AXE DES Y
194 '
196 FOR I=-180 TO 180 STEP 30
198 MOVE 4,I
200 DRAW -4,I
202 NEXT I
204 '
206 'SUBDIVISION DE L'AXE DES Y
208 '
210 MOVE -120,158
212 PRINT USING "####.##";150/FF;
214 '
216 TAGOFF
218 '
220 'DELIMITER LA SECTION EXAMINEE
```

```
222 '
224 MOVE -DEFI*FF,-DEFI*FF
226 DRAWR 2*DEFI*FF,0,2
228 DRAWR 0,2*DEFI*FF
230 DRAWR -2*DEFI*FF,0
232 DRAWR 0,-2*DEFI*FF
```

Description du programme :

100-102 Après fixation du MODE 1, on définit une flèche vers le haut sous le code de caractère 255 car le jeu de caractères n'offre pas de caractère approprié. Ce caractère sera utilisé lors du dessin du système de coordonnées.

104-118 *Quel domaine examiner*
Cette partie du programme demande à l'utilisateur d'indiquer quel domaine de définition il veut faire examiner et avec quel facteur d'agrandissement le programme devra travailler. Le facteur d'agrandissement est multiplié par 30 pour adapter le facteur à la graduation du système de coordonnées.

Après que ces deux valeurs aient été entrées, l'écran est vidé et le système de coordonnées y est dessiné (ligne 118).

120-128 *Dessin du système de coordonnées*
Pour préparer le dessin du système de coordonnées, la sortie de texte est adaptée au curseur graphique (TAG) et l'origine des coordonnées est fixée au centre de l'écran.

130-146 *Axe des X*
La partie de programme AXE DES X dessine une ligne horizontale au milieu de l'écran, avec le crayon de couleur 1. A l'extrémité droite de la ligne, on marque une pointe de flèche et le titre "x". Cette ligne représente l'axe des X du système de coordonnées à angle droit.

148-160 *Graduation de l'axe des X*
L'axe des X ainsi dessiné est subdivisé à l'aide de petits traits verticaux espacés tous les 30 points d'image.

162-170 *Marquage de l'axe des X*
La valeur correspondant, dans le système de coordonnées, au cinquième trait de graduation, sur la partie positive de l'axe des X, y est écrite en tenant compte du facteur FF d'agrandissement.

172-188 *Axe des Y*
La partie AXE DES Y du programme dessine une ligne verticale au milieu de l'écran. On fixe sur l'extrémité supérieure de la ligne la pointe de flèche, qui a été définie sous le code de caractère 255, ainsi que le titre "f(x)". Cette ligne verticale représente l'axe des Y du système de coordonnées.

190-202 *Graduation de l'axe des Y*
L'axe des Y ainsi dessiné est subdivisé à l'aide de petits traits horizontaux espacés tous les 30 points d'image.

204-216 *Marquage de l'axe des Y*
La valeur correspondant, dans le système de coordonnées, au cinquième trait de graduation, sur la partie positive de l'axe des Y, y est écrite en tenant compte du facteur FF d'agrandissement.

Une fois que le système de coordonnées a été entièrement dessiné, gradué et intitulé, la sortie de texte est redirigée sur la position du curseur de texte (TAGOFF).

218-232 *Marquer une section*
La section du système de coordonnées qui a été choisie pour la sortie par l'utilisateur est marquée par un cadre avec le crayon de couleur 2.

Les coins du cadre sont calculés par symétrie par rapport à la valeur entrée pour DEFI, multipliée par le facteur d'agrandissement FF, pour les quatre quarts du système de coordonnées.

6.2 FONCTIONS LINEAIRES

Vous vous souvenez certainement de la figure 6, qui représentait le graphe de la fonction f(x)=2*x (x appartenant à {1,2,3,4,5}). Nous avions pris ce graphe de fonction comme exemple pour exposer le principe de la représentation de fonctions mathématiques dans le système de coordonnées cartésien. Comme notre objectif est de dessiner des graphes de fonctions sur le CPC AMSTRAD, nous allons reprendre cette fonction dont vous connaissez déjà l'illustration pour faire un premier essai.

Pour dessiner le graphe de fonction, nous partirons du programme de dessin d'un système de coordonnées présenté au chapitre 6.1. Il faudra ajouter quelques lignes à ce programme pour représenter sur l'écran le graphe de la fonction f(x)=2*x.

Une fois que le système de coordonnées figure sur l'écran, les lignes de programme à ajouter devront simplement se charger de dessiner la fonction. Comme la fonction ne possède que cinq éléments dans son domaine de définition, tous les éléments du domaine de définition pourront être produits avec la boucle FOR-TO-NEXT. Cette boucle est parcourue exactement cinq fois, de 1 à 5, et la variable de comptage de la boucle prend successivement la valeur de tous les éléments du domaine de définition.

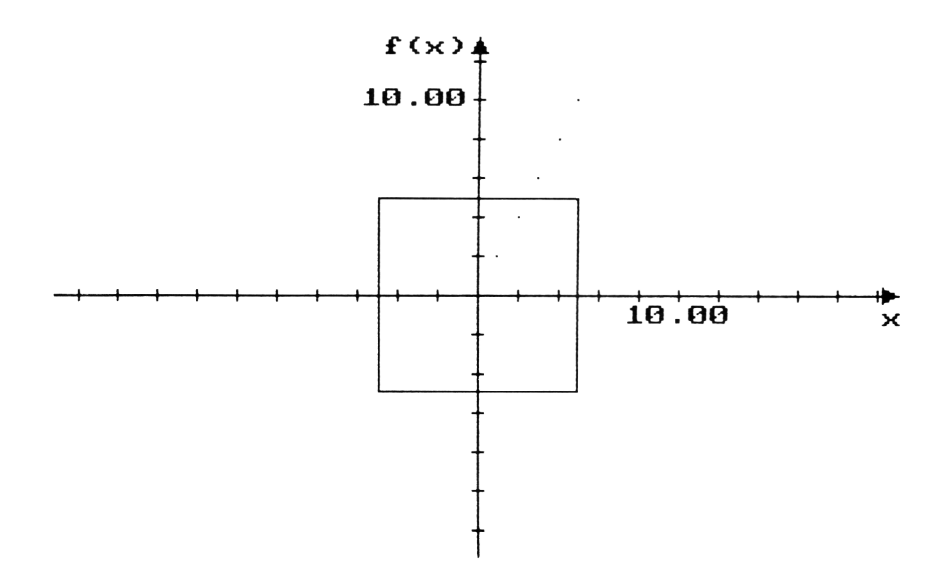
A partir des valeurs du domaine de définition, on peut calculer les valeurs du domaine de valeurs à partir de l'équation d'affectation.

Pour réaliser enfin le graphe de la fonction à la même échelle que le système de coordonnées, les valeurs X du domaine de définition et les valeurs Y du domaine de valeur doivent être multipliées par le facteur FF d'agrandissement.

```
236 'DESSIN DE LA FONCTION f(x)=2*x (x PARMI {1,2,3,4,5})
238 '
240 FOR X=1 TO 5
```

```
242 Y=2*X
244 PLOT X*FF,Y*FF
246 NEXT X
```

Une fois que les lignes de programme précédentes auront été ajoutées au programme du chapitre 6.1, le programme complet ainsi constitué produira la sortie écran que présente la copie d'écran 13. Nous avons entré 5 comme domaine de définition et 0,5 comme facteur d'agrandissement pour que le graphe de la fonction soit entièrement visible sur l'écran.



Copie d'écran 13 : **Graphe de la fonction f(x)=2*x (x parmi {1;2;3;4;5}) Domaine de définition 5 Facteur d'agrandissement 5**

La partie de programme que nous vous avons présentée vous fournit un graphe de fonction dans le cas particulier où la fonction est définie pour un nombre limité d'arguments mais elle restera totalement en rade s'il s'agit de dessiner la fonction pour le domaine de définition constitué par tous les nombres réels.

Les nombres réels sont en effet caractérisés par le fait qu'il existe toujours un troisième de ces nombres entre deux quelconques d'entre eux. Il est donc impossible dans ce cas de générer tous les éléments du domaine de définition dans une boucle comme c'était le cas dans l'exemple précédent.

Il nous faut donc trouver une méthode qui nous permette de dessiner le graphe de la fonction f(x)=2*x définie pour tous les x appartenant à l'ensemble des nombres réels. Nous pouvons ici tirer parti d'une particularité de cette fonction, la fonction f(x)=2*x appartient au groupe des fonctions linéaires.

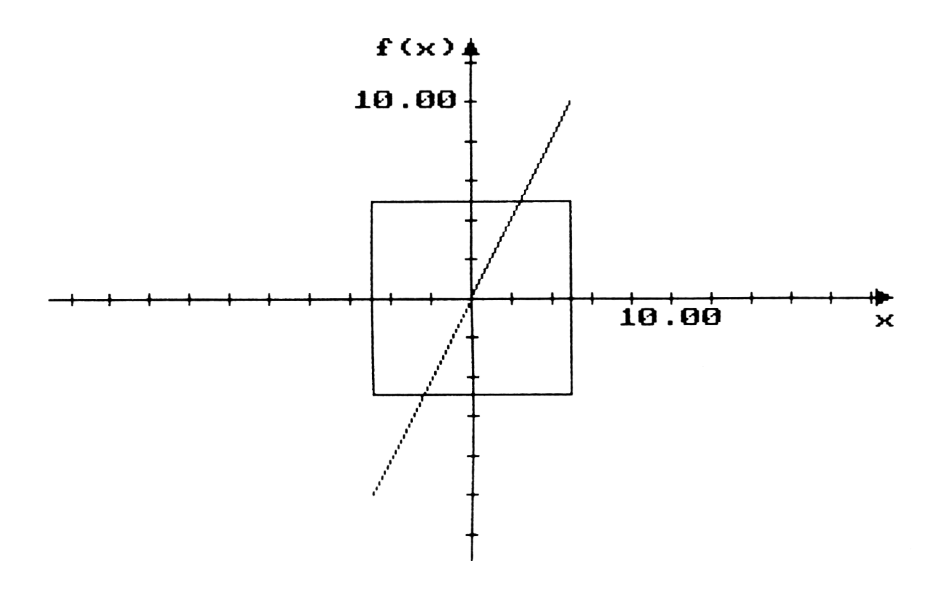
Le graphe de fonction des fonctions linéaires est en effet, par définition, toujours une droite. Une droite a la propriété, bien connue de tous ceux qui ont des notions de géométrie, de pouvoir être définie de manière non équivoque par deux points quelconques situés sur cette droite. Cela signifie qu'il suffit de connaître deux points situés sur une droite, de les relier entre eux et de prolonger la ligne créée à l'infini, pour dessiner une droite.

On peut mettre à profit cette propriété des fonctions linéaires lorsqu'on veut produire leurs graphes sur un ordinateur. Il suffit en effet de calculer les valeurs de fonction correspondant aux limites haute et basse du domaine de définition et on disposera des coordonnées de deux points que l'ordinateur pourra aisément relier entre eux.

Voici donc maintenant comment compléter le programme du chapitre 6.1 pour qu'il puisse dessiner le graphe de la fonction f(x)=2*x pour n'importe quel domaine de définition.

```
236 'DESSIN DE LA FONCTION f(x)=2*x
238 '
240 X=-DEFI
242 Y=2*X
244 MOVE X*FF,Y*FF
246 '
248 X=DEFI
250 Y=2*X
252 DRAW X*FF,Y*FF
```

La copie d'écran 14 montre que la méthode consistant à tirer le graphique complet de deux points du graphe de la fonction donne les résultats attendus. Il suffit naturellement de modifier l'équation de la fonction en lignes 242 et 250 pour représenter n'importe quelle autre fonction pourvu qu'il s'agisse d'une fonction linéaire.



Copie d'écran 14 : **Graphe de la fonction f(x)=2*x**
Domaine de définition 5
Facteur d'agrandissement 0,5

Le chapitre suivant vous présente un type de fonctions dont le graphe ne peut plus être dessiné à l'aide de la méthode que nous venons de vous présenter.

6.3 LES FONCTIONS QUADRATIQUES

Toute fonction qui ne répond pas à la définition des fonctions linéaires fera échouer la méthode présentée à la fin du chapitre 6.2 pour dessiner des graphes de fonction. Cela vient naturellement du fait que le graphe de ces fonctions n'est pas une droite et qu'il ne peut donc être déterminé de façon univoque par deux points. La fonction quadratique est un exemple de fonction de ce type.

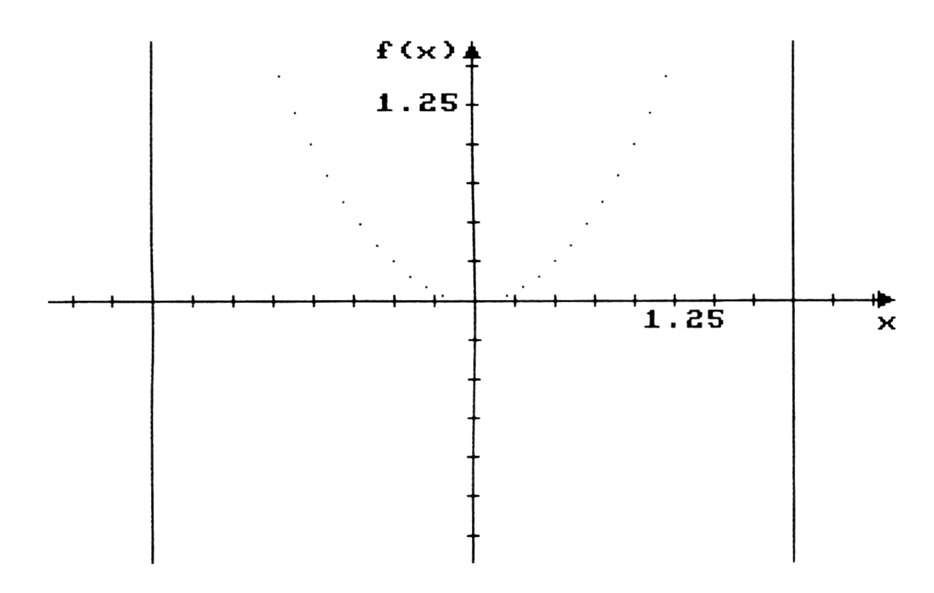
Les fonctions quadratiques, dont le nom vient de l'élévation à la puissance de leur argument, sont toujours représentées par des graphiques ayant la forme d'une parabole. On peut se représenter une parabole comme une sorte d'arc dont les branches se perdraient à l'infini. Cette simple description montre déjà bien qu'il s'agit là d'une figure totalement différente d'une fonction linéaire.

Pour pouvoir dessiner le graphe d'une fonction quadratique définie pour l'ensemble des nombres réels, il faut passer un compromis. Il serait en effet totalement impossible de prendre en compte tous les éléments du domaine de définition pour dessiner le graphe de la fonction. Le compromis consiste donc, très concrètement, à isoler des éléments du domaine de définition à intervalles réguliers, à calculer les valeurs de fonction correspondant à ces éléments et à dessiner le graphique à partir des coordonnées ainsi calculées. Le résultat de cette méthode ressemble beaucoup à la représentation de la fonction f(x)=2*x parmi {1;2;3;4;5}. C'est en effet le même principe qui est utilisé pour dessiner le graphe de la fonction. Mais il y a cependant une différence importante : dans le cas de la fonction linéaire définie pour cinq arguments, le graphe de fonction dessiné correspondait à la réalité alors que celui réalisé pour la fonction quadratique ne constitue qu'une approximation.

Les lignes imprimées ci-dessous complètent le programme de système de coordonnées de telle façon que le programme calculera une valeur de fonction par intervalles de 0,1 unités de graduation et dessinera à partir de là le graphe de la fonction f(x)=x*x, qu'on qualifie en mathématiques de parabole ordinaire.

```
236 'DESSIN DE LA FONCTION f(x)=x*x
238 '
240 FOR X=-DEFI TO DEFI STEP 0.1
242 Y=X*X
244 PLOT X*FF,Y*FF
246 NEXT X
```

La copie d'écran 15 vous présente l'image de la fonction f(x)=x*x obtenue avec ce programme pour un domaine de définition de -2 à 2, avec un agrandissement de 1 à 4 (facteur d'agrandissement = 4).



Copie d'écran 15 : **Graphe de la fonction f(x)=x*x**
Domaine de définition 2
Facteur d'agrandissement 4

L'intervalle entre les différents points formant le graphique peut être réduit à volonté; il suffit pour cela de diminuer le pas défini à la ligne 240 du complément du programme. Cela permet d'une part d'obtenir une image de la fonction beaucoup plus précise et l'on pourra même ne plus se rendre compte du compromis consistant à ne pas prendre tous les éléments du domaine de définition en compte

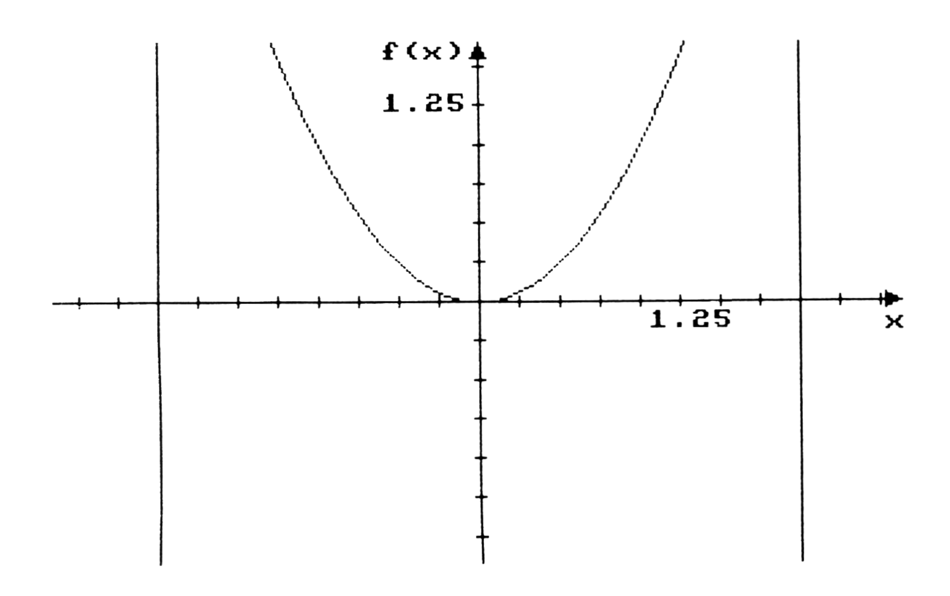
lors du dessin du graphique. Mais cette méthode présente également un inconvénient important. Plus l'intervalle entre les points sera faible et plus le programme travaillera lentement. Il risque même de travailler beaucoup plus lentement que votre patience ne vous permettrait de le supporter.

La solution peut nous être fournie ici par l'emploi combiné des deux méthodes présentées jusqu'à présent, c'est-à-dire le calcul de points isolés puis le tracé d'une ligne entre deux points. Nous connaissons déjà chacune de ces deux méthodes mais la combinaison des deux nous permettra de dessiner n'importe quelle fonction avec n'importe quel degré de précision. La précision sera définie ici par le nombre de valeurs de fonctions calculées et elle sera donc inversement proportionnelle à la vitesse d'exécution du programme.

```
236 'DESSIN DE LA FONCTION f(x)=x*x
238 '
240 X=-DEFI
242 Y=X*X
244 PLOT X*FF,Y*FF
246 '
248 FOR X=-DEFI TO DEFI STEP 0.1
250 Y=X*X
252 DRAW X*FF,Y*FF
254 NEXT X
```

Le fonctionnement du programme est très simple : on calcule la valeur de fonction correspondant à la valeur limite inférieure du domaine de définition. Cette valeur sera prise comme point de départ pour tracer la première ligne. On calculera ensuite toutes les autres valeurs de fonction jusqu'à la limite supérieure du domaine de définition, à l'intérieur d'une boucle FOR-TO-NEXT, d'après l'intervalle entré par l'utilisateur. Chaque fois qu'une nouvelle valeur de fonction aura été calculée, sa position sur l'écran sera calculée par multiplication par le facteur d'agrandissement. On tracera ensuite une ligne reliant le point précédent à ce nouveau point.

Le résultat obtenu, la parabole ordinaire, a maintenant une si belle apparence, avec un facteur d'agrandissement et un domaine de définition identiques à ceux de la copie d'écran 15, qu'aucune amélioration n'est plus nécessaire.



Copie d'écran 16 : **Graphe de la fonction f(x)=x*x**
Domaine de définition 2
Facteur d'agrandissement 4

6.4 LE PLOTTER DE FONCTION

Nous allons maintenant essayer de réutiliser tout ce que nous avons découvert dans les pages précédentes dans un programme d'application complet, le plotter de fonction. Ce programme nous permettra de représenter sur l'écran n'importe quelle fonction avec le degré de précision que nous voudrons. L'utilisateur pourra choisir librement le domaine de définition examiné, l'agrandissement employé mais aussi les intervalles entre les différentes valeurs de fonction calculées. Cette option supplémentaire permettra à l'utilisateur de se faire rapidement une idée de la forme générale du graphe de la fonction ou bien au contraire de faire dessiner la fonction avec le maximum de précision autorisée par la résolution de l'écran.

Une remarque préliminaire avant d'en venir à l'étude de notre plotter de fonction : le programme de plotter de fonction commence par un sous-programme contenant l'équation de la fonction. Nous avons choisi cet emplacement du programme pour que l'équation de la fonction puisse être facilement retrouvée si vous souhaitez la remplacer par une autre. Avant le lancement du programme, l'équation de la fonction dont le programme doit dessiner le graphe doit être entrée à la ligne 104. L'équation de la fonction doit toujours être écrite en respectant le format suivant : la variable X doit être l'argument de la fonction et Y la valeur de fonction de X. Le plotter de fonction ne pourra fonctionner correctement si vous ne respectez pas ce format.

Le sous-programme placé au début du plotter de fonction provoquera le message d'erreur "Unexpected RETURN in 108" si vous essayez de lancer le programme avec "RUN". Comme nous l'indiquons dans le listing, le plotter de fonction doit être lancé avec "RUN 112". Les lignes placées avant la ligne 112 ne pourront plus nuire au bon déroulement du programme. Si vous n'aimez pas devoir chaque fois taper "RUN 112" au lieu du simple "RUN", il vous suffit bien sûr d'ajouter au programme la ligne

```
10 GOTO 112
```

Une fois que vous avez lancé le programme comme il convient, vous voyez apparaître le titre du programme et le programme demande alors à l'utilisateur de donner trois indications :

1. Quel domaine de définition
2. Quel facteur d'agrandissement
3. Degré de précision

Les deux premières indications ne devraient pas vous poser de problème si vous avez lu attentivement le chapitre jusqu'ici. La troisième indication concerne l'intervalle entre les points qui devront être calculés. Il faut essayer de toujours choisir cette indication de façon à ce que, en tenant compte du facteur d'agrandissement, soient calculés tous les points image qui sont déterminants pour le graphe de la fonction. Une précision supérieure à celle autorisée par la résolution de l'écran n'a aucun intérêt car elle n'améliorera pas le résultat.

Une fois que le plotter de fonction a obtenu de l'utilisateur les indications nécessaires, le programme vide l'écran et commence à dessiner, graduer et intituler le système de coordonnées en MODE 1. Le cadre marquant la section sélectionnée est dessiné dans le système de coordonnées comme à l'habitude.

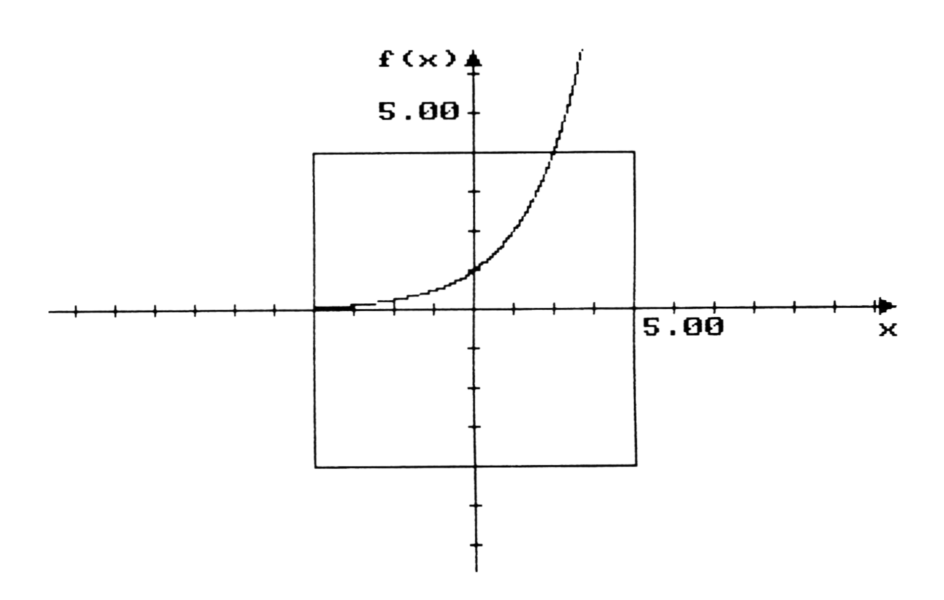
Une fois tous ces préparatifs effectués, le plotter de fonction commence à calculer les valeurs de fonction. Il indique alors au fur et à mesure par des points rouges sur l'axe des X les valeurs correspondant aux fonctions de valeurs calculées. Cet affichage n'indique cependant pas seulement la position des valeurs du domaine de définition pour lesquelles sont calculés des éléments du domaine de valeur. Elle permet en effet également de se rendre compte si le degré de précision choisi était le bon. S'il reste en effet des points vacants entre les points rouges, c'est que l'on n'a pas encore atteint la précision maximum. Les points rouges permettent également de rechercher la précision idéale en tâtonnant en sens inverse. Si vous remarquez en effet que la fonction est dessinée trop lentement, ce qui indique indubitablement que la précision choisie est exagérée, vous pouvez diminuer le degré de précision. Les points rouges vous permettront alors de contrôler que la précision obtenue reste suffisante.

Les points rouges vous donnent cependant encore une indication importante. Comme le travail commence toujours par la limite inférieure du domaine de définition pour se terminer à la limite supérieure, si les points rouges atteignent la limite supérieure c'est que le dessin de la fonction est terminé. Même si vous ne voyez aucun graphe de fonction sur l'écran. Dans ce cas, cela signifie simplement que l'utilisateur n'a pas choisi la bonne section pour l'examen de la fonction.

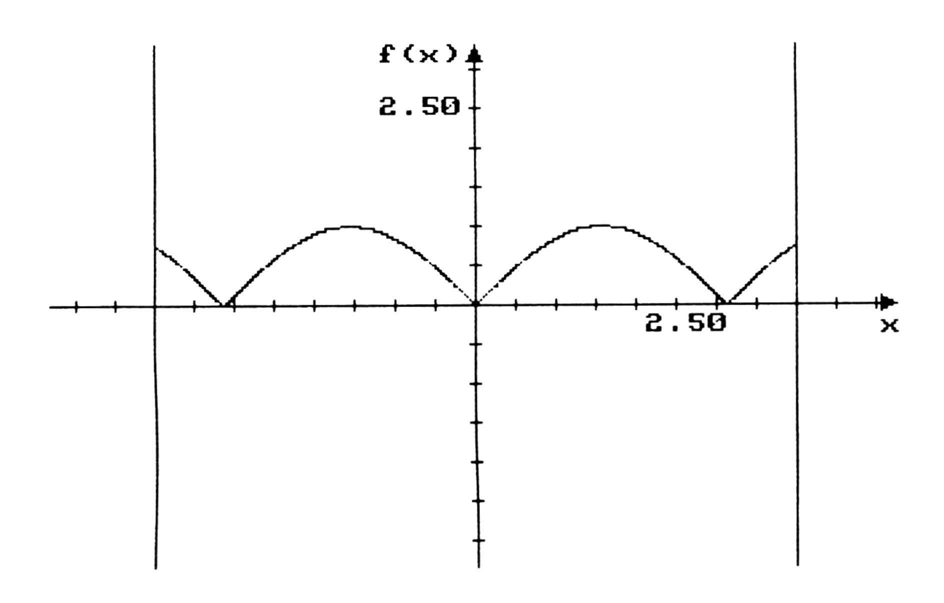
Le travail du plotter de fonction peut toujours être interrompu en appuyant sur une touche quelconque, ce qui fera revenir le programme à la question concernant le domaine de définition. C'est là également que le programme reviendra après avoir dessiné la fonction et après qu'une touche quelconque ait été actionnée. Le programme ne détruira donc jamais le graphique réalisé sans intervention de l'utilisateur. Le dessin peut donc être examiné tranquillement par l'utilisateur jusqu'à ce qu'il appuie sur une touche.

Si vous entendez votre CPC émettre un son à travers son haut-parleur pendant le dessin d'une fonction, cela signifie que le calcul d'une valeur de fonction a débouché sur un résultat qui ne peut être représenté par l'ordinateur. Le message d'erreur n'interrompt cependant pas le programme mais il vous signale que le graphe de fonction dessiné pose un problème.

Avant que vous ne commenciez à taper le listing du programme, permettez-nous de vous présenter encore deux graphes de fonctions avec les équations et les indications qui ont permis de les réaliser. Ces deux fonctions vous permettront de vérifier immédiatement que votre plotter de fonction fonctionne correctement.



Copie d'écran 17 : **Graphe de la fonction f(x)=2^x**
Domaine de définition 4
Facteur d'agrandissement 1
Degré de précision 0,01



Copie d'écran 18 : **Graphe de la fonction f(x)=ABS(SIN(X))**
Domaine de définition 4
Facteur d'agrandissement 2
Degré de précision 0,01

```
100 'ATTENTION! LE PLOTTER DE FONCTION DOIT ETRE LANCE AVEC
'RUN 112' CAR C'EST ICI QUE FIGURE L'EQUATION DE LA FONCTION
102 '
104 Y=COS(X)
106 '
108 RETURN
110 '
112 BORDER 0
114 INK 0,0
116 INK 1,26
118 INK 2,6
120 INK 3,11
122 '
124 SYMBOL 255,8,8,28,28,62,62,127,8
126 '
```

```
128 ON ERROR GOTO 340
130 '
132 'SORTIE DU TITRE DU PROGRAMME
134 '
136 MODE 2
138 ORIGIN 0,0
140 '
142 LOCATE 32,3
144 PRINT "PLOTTER DE FONCTION"
146 LOCATE 33,4
148 PRINT "JST  1.10.1986"
150 '
152 PLOT 16,383,1
154 DRAW 16,320
156 DRAW 623,320
158 DRAW 623,383
160 DRAW 16,383
162 '
164 'QUELLE ZONE DEVRA ETRE EXAMINEE
166 '
168 LOCATE 5,10
170 INPUT "QUEL DOMAINE DE DEFINITION VOULEZ-VOUS";DEFI
172 LOCATE 5,10
174 INPUT "QUEL FACTEUR D'AGRANDISSEMENT VOULEZ-VOUS";FF
176 FF=FF*30
178 LOCATE 5,10
180 INPUT "FIXEZ LE DEGRE DE PRECISION DU DESSIN";FH
182 '
184 'DESSIN DU SYSTEME DE COORDONNEES
186 '
188 MODE 1
190 TAG
192 ORIGIN 320,200
194 '
196 'AXE DES X
198 '
200 PLOT -320,0,1
202 DRAW 320,0
204 MOVER -16,6
206 PRINT CHR$(246);
208 MOVER -16,-16
210 PRINT "x";
```

```
212 '
214 'GRADUATION DE L'AXE DES X
216 '
218 FOR I=-300 TO 300 STEP 30
220 MOVE I,4
222 DRAW I,-4
224 NEXT I
226 '
228 'SUBDIVISION DE L'AXE DES X
230 '
232 MOVE 78,-8
234 PRINT USING "####.##";150/FF;
236 '
238 'AXE DES Y
240 '
242 MOVE 0,-200
244 DRAW 0,200
246 MOVER -8,-2
248 PRINT CHR$(255);
250 MOVER -80,0
252 PRINT "f(x)";
254 '
256 'GRADUATION DE L'AXE DES Y
258 '
260 FOR I=-180 TO 180 STEP 30
262 MOVE 4,I
264 DRAW -4,I
266 NEXT I
268 '
270 'SUBDIVISION DE L'AXE DES Y
272 '
274 MOVE -120,158
276 PRINT USING "####.##";150/FF;
278 '
280 TAGOFF
282 '
284 'MARQUER LA SECTION EXAMINEE
286 '
288 MOVE -DEFI*FF,-DEFI*FF
290 DRAWR 2*DEFI*FF,0,2
292 DRAWR 0,2*DEFI*FF
294 DRAWR -2*DEFI*FF,0
```

```
296 DRAWR 0,-2*DEFI*FF
298 '
300 'DESSINER LE GRAPHE DE FONCTION
302 '
304 X=-DEFI
306 GOSUB 104
308 PLOT X*FF,Y*FF,3
310 '
312 FOR X=-DEFI TO DEFI STEP FH
314 GOSUB 104
316 DRAW X*FF,Y*FF,3
318 '
320 PLOT X*FF,0,2
322 MOVE X*FF,Y*FF
324 '
326 IF INKEY$<>"" THEN 336
328 '
330 NEXT X
332 '
334 IF INKEY$="" GOTO 334
336 CLS
338 GOTO 130
340 '
342 'TRAITEMENT DES ERREURS
344 '
346 TAGOFF
348 PRINT CHR$(7);
350 TAG
352 '
354 RESUME NEXT
```

Description du programme :

100-108 Le plotter de fonction commence au tout début par un sous-programme contenant l'équation de la fonction. Le fait d'avoir placé l'équation tout au début du programme vous permettra de la retrouver facilement si vous voulez la modifier. Si vous lancez le plotter de fonction avec "RUN", vous provoquez le message d'erreur "Unexpected RETURN in 108".

Le programme doit donc être lancé avec "RUN 112" comme indiqué dans le listing, à moins que vous ne trouviez encore plus simple d'ajouter une ligne : 10 GOTO 112, auquel cas le programme pourra être lancé simplement avec "RUN".

112-128 Le programme fixe la couleur du cadre et les couleurs des crayons 0 à 3. Comme le générateur de caractères ne dispose pas d'un caractère approprié, nous définissons une flèche vers le haut sous le code 255. Ce caractère sera utilisé lors du dessin du système de coordonnées.

On ne peut exclure que les calculs effectués pendant le travail du plotter de fonction provoquent des erreurs (par exemple dépassements du domaine autorisé). Pour que ces erreurs ne provoquent pas une interruption du programme à la suite d'un message d'erreur, on fixe avec ON ERROR GOTO ce qui devra être fait en cas d'erreur.

130-160 *Sortie du titre du programme*
Après que le MODE 2 et l'origine des coordonnées aient été fixés, le titre du programme est sorti sur l'écran. Un cadre est dessiné autour de ces affichages.

162-180 *Quel domaine examiner*
Cette partie du programme demande à l'utilisateur d'indiquer quel domaine de définition il veut faire examiner, avec quel facteur d'agrandissement et avec quel intervalle de progression le programme devra travailler. L'intervalle est défini par l'entrée de FH. Toutes les indications de l'utilisateur peuvent être exploitées immédiatement par le programme à part le facteur d'agrandissement qui est multiplié par 30 pour adapter le dessin de la fonction à la graduation du système de coordonnées.

182-192 *Dessin du système de coordonnées*
Pour préparer le dessin du système de coordonnées, on fixe le MODE 1, la sortie de texte est adaptée au curseur graphique (TAG) et l'origine des coordonnées est fixée au centre de l'écran.

194-210 *Axe des X*
La partie de programme AXE DES X dessine une ligne horizontale au milieu de l'écran, avec le crayon de couleur 1. A l'extrémité droite de la ligne, on marque une pointe de flèche et le titre "x". Cette ligne représente l'axe des X du système de coordonnées à angle droit.

212-224 *Graduation de l'axe des X*
L'axe des X ainsi dessiné est subdivisé à l'aide de petits traits verticaux espacés tous les 30 points d'image.

226-234 *Marquage de l'axe des X*
La valeur correspondant, dans le système de coordonnées, au cinquième trait de graduation, sur la partie positive de l'axe des X, y est écrite en tenant compte du facteur FF d'agrandissement.

236-252 *Axe des Y*
La partie AXE DES Y du programme dessine une ligne verticale au milieu de l'écran. On fixe sur l'extrémité supérieure de la ligne la pointe de flèche, qui a été définie sous le code de caractère 255, ainsi que le titre "f(x)". Cette ligne verticale représente l'axe des Y du système de coordonnées.

254-266 *Graduation de l'axe des Y*
L'axe des Y ainsi dessiné est subdivisé à l'aide de petits traits horizontaux espacés tous les 30 points d'image.

268-280 *Marquage de l'axe des Y*
La valeur correspondant, dans le système de coordonnées, au cinquième trait de graduation, sur la partie positive de l'axe des Y, y est écrite en tenant compte du facteur FF d'agrandissement.

Une fois que le système de coordonnées a été entièrement dessiné, gradué et intitulé, la sortie de texte est redirigée sur la position du curseur de texte (TAGOFF).

282-296 *Marquer une section*
La section du système de coordonnées qui a été choisie pour la sortie par l'utilisateur est marquée par un cadre avec le crayon de couleur 2. Les coins du cadre sont calculés par symétrie par rapport à la valeur entrée pour DEFI, multipliée par le facteur d'agrandissement FF, pour les quatre quarts du système de coordonnées.

298-338 *Dessin du graphe de la fonction*
Comme le graphe de la fonction n'est pas dessiné avec l'instruction BASIC PLOT mais avec DRAW, pour obtenir un tracé continu, il faut calculer le point de départ de la première ligne. C'est ce dont se chargent les lignes 304 à 308 qui sélectionnent également le crayon de couleur 3.

Le graphe de la fonction est dessiné dans la boucle FOR-TO-NEXT des lignes 312 à 330. La boucle va de la limite négative à la limite positive du domaine de définition, le pas de progression étant défini par la variable FH dont la valeur a été fixée au début du programme en fonction de l'entrée de l'utilisateur. Pendant toute l'opération de dessin, on surveille si une touche est actionnée, ce qui entraînerait l'interruption du dessin et un saut au début du programme.

Les lignes 320 et 322 de la boucle FOR-TO-NEXT sont chargées d'indiquer à quelles valeurs du domaine de définition le plotter de fonction en est actuellement pour le calcul des valeurs correspondantes. Cette indication est fournie à l'utilisateur sous forme d'une série de points rouges affichés, le long de l'axe des X, de la limite négative du domaine de définition à sa limite positive.

Une fois que la boucle FOR-TO-NEXT a été entièrement exécutée, le dessin réalisé demeure sur l'écran jusqu'à ce qu'une touche quelconque soit actionnée. Dès qu'on appuie sur une touche, le programme vide l'écran et revient à la ligne 130 où d'autres indications peuvent être entrées par l'utilisateur pour le dessin du graphe de la fonction.

340-Fin *Traitement des erreurs*
Si une erreur apparaît au cours du calcul des valeurs de fonction (dépassement du domaine de valeurs autorisées), le programme saute automatiquement ici, il produit un signal sonore d'avertissement et il continue ses calculs avec la valeur suivante.

7. Le graphisme en trois dimensions

Beaucoup d'entre vous se souviennent certainement encore des films de science-fiction du cycle de la "Guerre des étoiles". De nombreux spectateurs de cinéma ont été fascinés par les images en trois dimensions créées par ordinateur que l'on pouvait voir dans ces films. Ces images avaient été créées par Georges Lucas en 1977 en collaboration avec le géant actuel des effets spéciaux, "Industrial Light and Magic". La plupart des images qu'on pouvait voir sur les écrans étaient produites avec une technique fondée sur la "représentation vectorielle". Les images de synthèse ainsi créées se composent uniquement de lignes symbolisant les contours d'un corps. Le résultat d'une représentation de ce type est une armature de fils qu'on appelle wire frame en anglais technique.

Mais comment parvient-on à obtenir de telles armatures par le calcul? Comme son nom l'indique, la représentation vectorielle a recours à ce qu'on appelle des vecteurs. Un vecteur est un trait orienté dans l'espace. On peut dire également que c'est une flèche invisible dirigée vers un point quelconque d'un univers imaginaire. C'est justement cette propriété que de nombreux programmeurs exploitent lorsqu'ils veulent développer une image en 3 dimensions. L'armature produite par les vecteurs est en effet à la base de toute animation en trois dimensions définie mathématiquement. Tous les modèles complexes à superficie pleine ne sont rien d'autre que l'extrapolation logique d'une armature de fils générée grâce à la technique des vecteurs.

On utilise les dessins informatiques animés tridimensionnels de plus en plus de nos jours dans la publicité, le cinéma ou encore dans le domaine scientifique. Il y a par exemple aux USA et en France des studios graphiques spécialisés dans la production de films électroniques ainsi réalisés. On y produit des images de synthèse qui dépassent souvent nos facultés d'abstraction et d'imagination grâce à des installations informatiques gigantesques et à une résolution graphique atteignant en moyenne 1024 fois 1024 points. Les séquences animées produites à l'aide de ces ordinateurs frappent toujours par leur réalisme et leur perfection des détails vraiment incroyables. Le prix de revient de ce travail de qualité est autour de 4000 dollars la seconde !

Revenons maintenant à notre CPC. La représentation vectorielle est également utilisée sur les ordinateurs domestiques comme moyen d'expression plastique et visuelle dans les logiciels commerciaux. Des jeux électroniques tels qu'ELITE (marque déposée de Acornsoft Ltd.) par exemple tirent leur puissant réalisme d'une armature se modifiant en permanence à l'horizon.

Les animations à trois dimensions de ce type ne sont pas réservées uniquement aux sociétés d'édition de logiciels commerciaux. La technique des vecteurs permet d'obtenir même en BASIC des effets tout à fait surprenants. Les rotations sans à coup d'objets définis par programme ne sont naturellement possibles qu'en langage machine. Nous allons toutefois vous présenter tous les exemples suivants de programmes en BASIC. En effet, la programmation machine des algorithmes nécessaires est tellement compliquée qu'il faudrait leur consacrer un ouvrage entier et d'autre part l'interpréteur BASIC présente le grand avantage de vous permettre plus facilement de faire des tests et des essais. Vous trouverez à la fin de ce chapitre un programme d'animation en 3 dimensions complet, écrit en BASIC, avec un éditeur et un mode démonstration intégrés. Comme nous ne voulons cependant pas abandonner ceux d'entre vous qui sont des programmeurs du Z80, nous vous donnerons également quelques conseils pour convertir le programme en langage machine.

Pour que vous soyez à même à l'avenir de réaliser par vous-même un programme 3 dimensions vous convenant parfaitement, voici tout d'abord un peu de théorie. Mais ne craignez rien! Ces réflexions théoriques ne dégénéreront pas en cours complet de "géométrie analytique". Nous nous en tiendrons aux faits essentiels et nous expliquerons quelques formules toute prêtes de façon à ce que vous puissiez parvenir le plus rapidement possible au succès escompté.

7.1 LE SYSTEME DE COORDONNEES EN TROIS DIMENSIONS

Comme nous l'indiquions dans le chapitre précédent, le système de coordonnées à deux dimensions se compose de deux axes se coupant à angle droit qu'on appelle l'axe des X et l'axe des Y et qui représentent la "hauteur" et la "largeur". Dans le système de coordonnées tridimensionnel, une troisième dimension vient s'ajouter, la profondeur. Elle est représentée par ce qu'on appelle l'axe des Z. L'illustration suivante illustre cette définition :

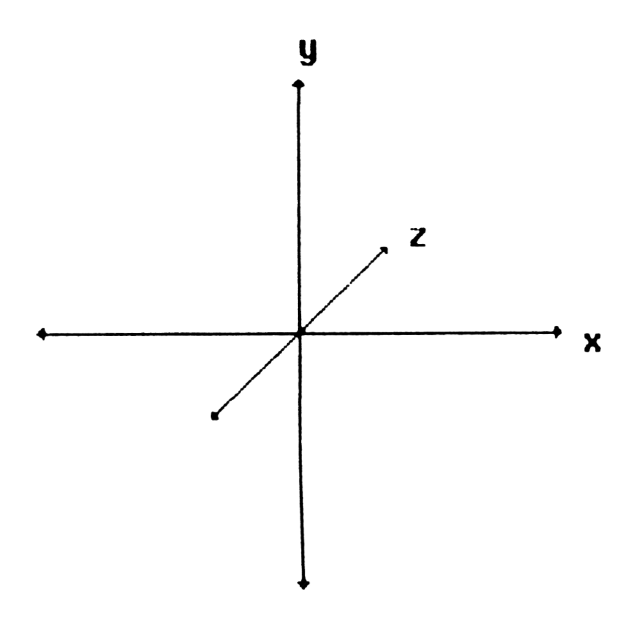


Figure 9 : Le système de coordonnées tridimensionnel

Avec un système de coordonnées tridimensionnel, on peut définir n'importe quel point situé dans un espace déterminé à l'aide de ses coordonnées X, Y et Z. Si on veut par exemple définir un point situé au centre du système de coordonnées, les coordonnées de ce point pourront être définies à l'aide de la matrice P(0,0,0). Les trois éléments de la matrice représentent les coordonnées X, Y et Z. Il en résulte que chaque coordonnée définit la situation d'un corps par rapport à l'axe auquel elle est affectée. P(0,0,100) définira par exemple un point situé sur la partie de l'axe des Z dirigée vers l'observateur alors que P(0,0,-100) adressera la partie négative de ce même axe.

La taille de l'espace tridimensionnel dépend naturellement de la longueur des différents axes du système de coordonnées. Le plus simple pour adresser les coordonnées devant être définies sur ces axes est d'utiliser trois tableaux unidimensionnels (=trois vecteurs) que vous pouvez définir sans aucun problème avec l'instruction DIM de votre interpréteur BASIC.

La taille de chaque vecteur n'est dans ce cas limitée que par la place mémoire dont dispose votre ordinateur.

Exemple :

```
100 XN=300:'Nombre maximal de coordonnées X
110 YN=300:'Nombre maximal de coordonnées Y
120 ZN=300:'Nombre maximal de coordonnées Z
130 DIM X(XN), Y(YN), Z(ZN)
```

Il y a trois façons de définir un objet dans notre système de coordonnées.

1. Le contenu entier de la superficie d'un objet est transféré dans le système de coordonnées.

2. La définition d'un objet se limite aux coordonnées de tous ses angles et des liaisons entre ces angles.

La première solution a l'inconvénient de nécessiter l'écriture dans nos vecteurs de très nombreux points pour la définition d'un seul objet. Une rotation de l'objet entraînerait de ce fait un travail de calcul extrêmement élevé. Enfin cette méthode rend pratiquement impossible toutes modifications ultérieures de l'objet.

La seconde possibilité convient par contre parfaitement à l'objectif que nous recherchons. Comme les vecteurs ne contiennent que les angles d'un objet, les opérations de calcul qui permettront d'obtenir plus tard l'animation pourront se limiter à ces vecteurs. Cette méthode ne permet pas toutefois de représenter l'objet entièrement. Pour y parvenir, il nous faut avoir recours à une petite astuce. Il suffit en effet de relier tous les angles par des lignes. Pour cela, il faut que soit possible en permanence l'accès aux données définissant les liens entre les angles.

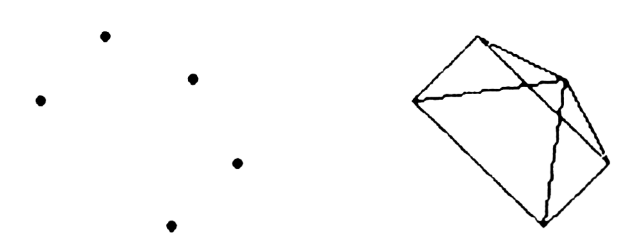


Figure 10 : **Angles/angles reliés**

Notre programme tridimensionnel devra donc disposer dans sa version définitive non seulement des trois vecteurs mais aussi d'un bloc de données supplémentaire qui définira les liens entre les différentes coordonnées. Ce bloc de données pourra être produit à l'aide d'une matrice à deux dimensions. Comme il n'est pas possible de définir d'avance la taille de cette matrice, il est préférable de choisir un nombre élevé pour dimensionner le tableau correspondant à ce bloc de données. Le premier paramètre de l'instruction

```
DIM V(255,1)
```

indiquera dans cet exemple le nombre de liens. Le second paramètre permet de réserver chaque fois deux paramètres (0 et 1) pour les numéros d'angle à relier entre eux.

Une matrice à deux dimensions et trois vecteurs suffisent donc à la définition complète d'un objet dans l'espace tridimensionnel. Mais comment peut-on ensuite le projeter sur l'écran ?

Les mathématiques offrent deux solutions possibles :

- la projection parallèle et
- la projection centrale.

Ces deux modes de projection nous seront utiles pour l'objectif qui est le nôtre. C'est en fonction des domaines particuliers d'application du graphisme tridimensionnel et de vos goûts personnels que vous devrez par la suite donner, suivant les cas, la préférence à l'un ou l'autre mode. Nous allons en tout cas vous présenter le fonctionnement des deux méthodes en vous expliquant les avantages et les inconvénients inhérents à chacune.

7.2 LA PROJECTION PARALLELE

Ce mode de projection constitue la plus simple des deux méthodes. Elle consiste à représenter les objets dans ce qu'on appelle la perspective parallèle. Les objets projetés sur l'écran au moyen de la perspective parallèle ne sont reliés à aucun point de fuite; c'est-à-dire qu'il n'y a pas, dans l'exemple qui nous intéresse, de point défini vers lequel tendraient toutes les lignes. Il en résulte que la taille d'un corps ne peut plus être perçue dans un espace tridimensionnel.

La projection parallèle a cependant aussi des avantages. Comme elle peut être réalisée sans une trop grande charge de travail dans des programmes système tridimensionnels, elle permet d'obtenir même en BASIC des vitesses de projection remarquables. Par ailleurs des objets réalisés en projection avec perspective parallèle permettent d'atteindre des effets plastiques même avec une résolution faible de l'écran (par exemple en MODE 0).

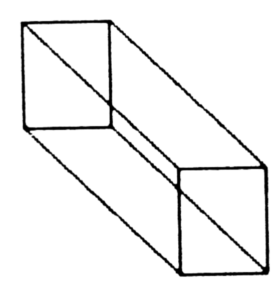


Figure 11 : **Exemple de projection parallèle**

Mais venons-en maintenant à l'étude de la méthode de projection elle-même. Comme l'écran ne possède que deux dimensions, les dessins ne peuvent y être sortis qu'en deux dimensions. Pour obtenir un effet de "simulation" des trois dimensions, le dessin doit subir une distorsion sur l'écran. Cela signifie dans le cas qui nous intéresse qu'il faut manipuler les coordonnées X et Y des points de départ et des points de destination de toutes les lignes de l'objet défini avec les paramètres Z qui leur correspondent. Voici les formules qui permettent d'effectuer de telles transformations :

```
XP=X(V(I,J))+Z(V(I,J))*SIN(ALPHA)
YP=Y(V(I,J))+Z(V(I,J))*SIN(BETA)
```

Dans les deux cas, le produit de la coordonnée Z et de l'angle d'observation est additionné à l'élément du vecteur indexé par la matrice de relation V(I,J). Il est indiqué par les variables ALPHA et BETA. Pour que vous puissiez plus facilement vous faire une idée de l'effet de ces formules, nous allons vous présenter maintenant un petit programme de démonstration avec un objet déjà entièrement défini. Ses angles et les lignes de liaison sont stockés dans des blocs de DATA. Vous pouvez donc aisément les manipuler. Faites autant d'essais que vous le voudrez, avec différents angles de vision et différents objets. Vous verrez alors combien le graphisme en trois dimensions peut être fascinant.

```
10 '*********************************************************
20 '**                                                     **
30 '**  DEMONSTRATION DE PROJECTION PARALLELE              **
40 '**          7.9.'86 TAV                                **
50 '**                                                     **
60 '*********************************************************
70 '
80 '
100 'INITIALISATION
110 '
120 MODE 1
130 DEG
140 ORIGIN 320,200
150 DEFINT A-Z
160 DIM X(255),Y(255),Z(255),V(255,1)
170 '
```

```
180 'LECTURE DES COORDONNEES
190 '
200 READ J
210 FOR I=0 TO J:READ X(I),Y(I),Z(I):NEXT
220 '
230 'LECTURE DES LIENS
240 '
250 READ VB
260 FOR I=0 TO VB:READ V(I,0),V(I,1):NEXT
270 '
280 'ENTREE DE L'ANGLE DE VISION
290 '
300 CLS
310 PRINT"ENTREZ L'ANGLE DE VISION SVP"
320 PRINT:PRINT
330 INPUT"ALPHA (0-360)";ALPHA
340 INPUT"BETA (0-360)";BETA
350 CLS
360 '
370 'BOUCLE DE PROJECTION
380 '
390 FOR I=0 TO VB
400 J=0:GOSUB 510
410 MOVE XP,YP
420 J=1:GOSUB 510
430 DRAW XP,YP
440 NEXT
450 '
460 LOCATE 8,25:PRINT"VEUILLEZ FRAPPER UNE TOUCHE!";
470 IF INKEY$="" THEN 470 ELSE 300
480 '
490 'CONVERSION EN 3 D
500 '
510 XP=X(V(I,J))+Z(V(I,J))*SIN(ALPHA)
520 YP=Y(V(I,J))+Z(V(I,J))*SIN(BETA)
530 RETURN
540 '
550 'COORDONNEES
560 '
570 DATA 7:'NOMBRE DES COORDONNEES MOINS 1
575 '
580 DATA -50,-50,-50
```

```
590 DATA -50,50,-50
600 DATA 50,50,-50
610 DATA 50,-50,-50
620 DATA -50,-50,50
630 DATA -50,50,50
640 DATA 50,50,50
650 DATA 50,-50,50
660 '
670 'LIENS
680 '
690 DATA 13:'NOMBRE DES LIAISONS MOINS 1
695 '
700 DATA 0,1
710 DATA 1,2
720 DATA 2,3
730 DATA 3,0
740 DATA 4,5
750 DATA 5,6
760 DATA 6,7
770 DATA 7,4
780 DATA 0,4
790 DATA 1,5
800 DATA 2,6
810 DATA 3,7
820 DATA 0,2
830 DATA 1,3
```

Description du programme :

100-160 *Initialisation*
Sélection de la résolution graphique de 320 points sur 200, du calcul en degrés et dimensionnement de tous les vecteurs et matrices. Toutes les variables sont ensuite déclarées entières (DEFINT) et l'origine des coordonnées est placée au centre de l'écran.

170-260 *Lecture des données*
Les éléments en DATA sont chargés dans les tableaux d'angles et de liaisons. L'instruction READ placée avant chaque boucle permet de lire chaque fois le nombre d'éléments de données-1 qui seront chargés dans la variable-matrice correspondante par la boucle FOR-TO-NEXT placée sur la ligne suivante (J et VB).

270-350 *Définition de l'angle de vision*
L'angle de vision en degrés est chargé dans les variables ALPHA et BETA au moyen d'une instruction INPUT. L'écran est ensuite vidé.

360-470 *Boucle de projection*
Elle va chercher toutes les lignes d'un objet dans les tableaux et retransmet les indices des coordonnées de départ et de fin à la routine de conversion en 3 D (GOSUB 510). Les coordonnées X1, X2, Y1 et Y2 obtenues dans XP et YP sont utilisées pour le dessin des lignes, dans les lignes de programme 410 et 430.

Une fois l'image terminée, la routine en ligne 470 attend qu'une touche soit frappée. Une fois qu'une touche a été actionnée, retour à la routine de définition de l'angle d'observation.

480-530 *Conversion 3 D*
Cette routine calcule les positions X et Y d'un angle sur l'écran en perspective parallèle.

540-650 *Coordonnées*
Le premier élément DATA indique le nombre de lignes DATA suivantes -1.

660-Fin *Liaisons*
Le premier élément DATA indique le nombre de lignes DATA suivantes -1.

Si l'objet doit en plus être déplacé dans l'espace, toutes les coordonnées doivent subir un décalage approprié. Ce décalage peut être aussi bien positif que négatif. Après application de ce décalage (offset), nous obtenons les formules suivantes :

```
XP=X(V(I,J))+XOFF+(Z(V(I,J))+ZOFF)*SIN(ALPHA)
YP=Y(V(I,J))+YOFF+(Z(V(I,J))+ZOFF)*SIN(BETA)
```

Pour ramener un objet déplacé sur sa position de départ, il faut affecter à XOFF, YOFF et ZOFF la valeur 0.

7.2.1 Rotation dans l'espace

Nous ne sommes parvenus jusqu'ici qu'à déplacer sur l'écran un objet tridimensionnel et à l'observer sous différents angles. Bien que les résultats graphiques ainsi obtenus produisent une forte impression de trois dimensions, on peut encore améliorer le résultat. Une animation tridimensionnelle réaliste ne se conçoit pas en effet sans libre rotation dans l'espace. Pour que vous puissiez faire pivoter un objet autour d'un des trois axes, les coordonnées tridimensionnelles doivent être converties en coordonnées de rotation correspondantes. Cette conversion peut être obtenue pour chaque axe avec deux formules différentes.

Rotation sur l'axe des X :

```
YR=Y(V(I,J))*COS(ALPHA)+Z(V(I,J))*SIN(ALPHA)
ZR=-Y(V(I,J))*SIN(ALPHA)+Z(V(I,J))*COS(ALPHA)
```

Rotation sur l'axe des Y :

```
XR=X(V(I,J))*COS(BETA)+Z(V(I,J))*SIN(BETA)
ZR=-X(V(I,J))*SIN(BETA)+Z(V(I,J))*COS(BETA)
```

Rotation sur l'axe des Z :

```
XR=X(V(I,J))*COS(GAMMA)+Y(V(I,J))*SIN(GAMMA)
YR=-X(V(I,J))*SIN(GAMMA)+Y(V(I,J))*COS(GAMMA)
```

Les variables ALPHA, BETA et GAMMA doivent se voir affecter un angle de rotation avant que la routine de calcul ne soit appelée. A cet égard, il faut absolument tenir compte du fait que la rotation ne peut se faire simultanément sur tous les axes. Si c'est cependant ce que vous voulez, il faut que les données obtenues avec le premier groupe de deux formules soient réutilisées dans les deux équations suivantes.

7.3 LA PROJECTION CENTRALE

Bien que notre méthode de projection semble parfaite pour le moment, la rotation de l'objet a parfois un aspect peu réaliste. Pourquoi? En fait, lorsque nous observons un corps sous des angles et des distances différents, nous constatons qu'il y a une relation entre son éloignement par rapport à l'oeil humain et l'ensemble de son apparence extérieure. Ce phénomène est dû au fait que tous les contours d'un objet que nous percevons tendent à nos yeux vers un point défini de l'horizon qu'on appelle le point de fuite. C'est pourquoi un objet nous apparaît d'autant plus petit qu'il semble être proche de ce point de fuite fictif. C'est d'ailleurs justement cet effet, qu'on appelle aussi "track effect" en anglais technique, qui aide le cerveau humain à déterminer les dimensions d'un objet ainsi que son éloignement par rapport à l'observateur. C'est aussi ce principe qui est cause du manque de réalisme des méthodes de projection que nous avons utilisées jusqu'ici.

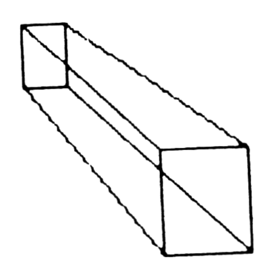


Figure 12 : Exemple de projection centrale

La projection centrale nous permet, contrairement à la projection parallèle, de définir dans notre système de coordonnées tridimensionnel un point de fuite fictif. Cette méthode est malheureusement extrêmement complexe et elle prend donc beaucoup de temps de calcul. Mais c'est au bout du compte, la seule possibilité pour produire une structure tridimensionnelle parfaite sur l'écran. Les formules suivantes vous permettent de projeter un objet en perspective centrale sur l'écran. Ne soyez surtout pas effrayé par le nombre et la complexité des formules nécessaires !

Nous vous présenterons plus loin une petite astuce qui permet de diminuer de façon importante le temps de calcul nécessaire.

```
P0 = COS(GAMMA)*COS(BETA)
P1 = SIN(GAMMA)*COS(BETA)
P2 =-SIN(BETA)
P3 =-SIN(GAMMA)*COS(ALPHA)+COS(GAMMA)*SIN(BETA)*SIN(ALPHA)
P4 = COS(GAMMA)*COS(ALPHA)+SIN(GAMMA)*SIN(BETA)*SIN(ALPHA)
P5 = COS(BETA)*SIN(ALPHA)
P6 = SIN(GAMMA)*SIN(ALPHA)+COS(GAMMA)*SIN(BETA)*COS(ALPHA)
P7 =-COS(GAMMA)*SIN(ALPHA)+SIN(GAMMA)*SIN(BETA)*COS(ALPHA)
P8 = COS(BETA)*COS(ALPHA)

ZP=1-(X(V(I,J)*P2+Y(V(I,J))*P5+Z(V(I,J))*P8+ZOFF)/FP
XP=(X(V(I,J)*P0+Y(V(I,J))*P3+Z(V(I,J))*P6+XOFF)/ZP
YP=(X(V(I,J)*P1+Y(V(I,J))*P4+Z(V(I,J))*P7+YOFF)/ZP
```

Paramètres en entrée :

ALPHA	= Angle de la rotation sur l'axe des X (0 à 360 degrés)
BETA	= Angle de la rotation sur l'axe des Y (0 à 360 degrés)
GAMMA	= Angle de la rotation sur l'axe des Z (0 à 360 degrés)
FP	= Point de fuite
V	= Matrice des lignes de liaison
I	= Numéro de ligne de liaison (0 à n-1)
J	= Index pour les points de départ et de fin d'une liaison (0 à 1)
X(n-1)	= Vecteur X
Y(n-1)	= Vecteur Y
Z(n-1)	= Vecteur Z

Paramètres en sortie :

XP	= Coordonnée X sur l'écran
YP	= Coordonnée Y sur l'écran

7.4 ASTUCES PERMETTANT UNE ANIMATION PLUS RAPIDE DE L'OBJET

Tables de fonctions trigonométriques

Comme le graphisme vectoriel tridimensionnel est produit à l'aide de fonctions d'angle, le temps de calcul nécessaire pour construire une seule image peut vite s'élever à plusieurs secondes. Cet effet secondaire particulièrement néfaste se produit surtout en liaison avec la perspective centrale. Pour que les temps de calcul forcément élevés qui en résultent puissent être ramenés à des proportions plus raisonnables, on peut se servir de tables de fonctions trigonométriques. Elles permettront d'éviter de devoir appeler, pendant la projection de l'objet, des routines de calcul de sinus et de cosinus qui risqueraient de prendre beaucoup de temps. Ce n'est qu'en ayant recours à ces tables que l'animation d'un objet sur les microordinateurs devient possible.

L'intégration de l'emploi des tables de sinus et de cosinus nécessaires est très simple. Il faut dimensionner à 360, avec l'instruction DIM, deux autres vecteurs dans le bloc d'initialisation de votre programme : SI! et CO!.

```
DIM SI!(360),CO!(360)
```

SI! représente ici la table des sinus et CO! celle des cosinus. La valeur de 360 correspond à 360 degrés. Il est conseillé de créer les deux tables immédiatement après cela. La ligne BASIC

```
FOR I=0 TO 360:SI!(I)=SIN(I):CO!(I)=COS(I):NEXT
```

se chargera de ce petit travail. Il suffit maintenant de remplacer dans la routine de calcul SIN et COS par SI! et CO! - c'est tout!

Désormais votre programme tridimensionnel n'aura plus jamais besoin de calculer une fonction trigonométrique. Si un calcul de ce type est requis, le programme ira automatiquement chercher la valeur nécessaire dans l'une des deux tables. Vous pourrez bientôt apprécier par vous-même le gain de vitesse que cela permet d'obtenir.

Animation sans tremblotement

L'effet tridimensionnel d'un dessin vectoriel est malheureusement fortement limité par la construction et le démantèlement des différentes images servant à simuler l'animation d'un objet. Avec des structures de grille complexes, cet effet peut même détruire complètement l'effet de mouvement. Ce problème peut être résolu lorsqu'on a la possibilité de travailler avec deux pages d'écran complètement indépendantes l'une de l'autre. Il faut pour cela réserver les 16 K de la seconde banque de mémoire (&4000 à &7FFF) avec l'instruction MEMORY et définir ce bloc de la mémoire comme page d'écran alternative.

Attention : Ceux qui possèdent un CPC 6128 peuvent naturellement obtenir le même effet avec les instructions de banque de mémoire ISCREENCOPY et ISCREENSWAP. Par souci de compatibilité, nous allons cependant vous présenter ici une méthode universelle. Elle tourne aussi vite qu'une routine réalisée avec les instructions SCREEN et elle tourne sur tous les ordinateurs CPC.

Après que la zone de la RAM qui va de &4000 à &7FFF ait été protégée avec l'instruction MEMORY, la séquence

```
OUT &BC00,12:OUT &BD00,16
```

vous permet d'activer la page écran alternative (&4000) et

```
OUT &BC00,12:OUT &BD00,52
```

de revenir à la page écran normale (&C000). Les deux séquences OUT effectuent une manipulation délibérée du contrôleur vidéo. C'est beaucoup plus efficace dans le cas qui nous occupe que d'utiliser un vecteur car ce dernier fixerait en plus toutes les fonctions graphiques sur la nouvelle zone de mémoire écran et c'est justement ce que nous voulons éviter.

Il ne nous faut plus maintenant qu'une petite routine machine qui puisse copier le contenu complet de l'écran de la page écran normale sur notre page écran alternative.

Le jeu d'instructions de l'unité centrale Z80 possède heureusement une instruction de déplacement de bloc, l'instruction LDIR qui se chargera de ce travail à notre place. Il suffit de l'initialiser de la façon suivante :

Le code Z80 :

```
LD HL,&C000        'Copier l'écran normal
LD DE,&4000        'dans l'écran alternatif
LD BC,&4000        'Copier un total de 16K
LDIR               'et ce maintenant!
RET                'Terminé!
```

La routine BASIC de chargement :

```
100 FOR I=0 TO 11:READ J:POKE &3FF4+I,J:NEXT
200 DATA &21,&00,&C0,&11,&00,&40,&01,&00,&40,&ED,&B0,&C9
```

Il suffit maintenant d'appeler les différentes étapes dans l'ordre suivant :

1. activer la page écran normale

2. copier avec l'instruction LDIR la page écran normale (&C000) dans la banque 1 (&4000)

3. activer la page écran alternative

4. vider l'écran normal

5. réaliser le dessin sur la page écran normale

6. répéter les étapes 1 à 5 jusqu'à ce que l'animation soit terminée

Il est aisé maintenant de réaliser une routine tenant compte de ces informations :

```
100 OUT &BC00,12:OUT &BD00,52
110 CALL LDIR
120 OUT &BC00,12:OUT &BD00,16
```

```
130 REM *** Animation ***
              .
              .
              .
999 GOTO 100
```

7.5 CPCs-WORLD

Nous allons maintenant vous présenter un programme complet permettant de réaliser du graphisme vectoriel tridimensionnel animé. Il vous permettra de déplacer et de faire tourner sur tous les axes du système de coordonnées un modèle d'armature que vous aurez défini. La méthode de projection choisie est la projection centrale. Le programme s'appelle "CPCs World" et comprend cinq sections différentes.

1. Le générateur d'objet
2. L'éditeur d'objet
3. L'éditeur d'animation
4. Le mode d'animation tridimensionnelle
5. Chargement, sauvegarde et suppression d'un objet

Comme cette présentation vous permet de le supposer, nous avons attaché une importance particulière, lors du développement du programme, à ce que la réalisation d'une séquence d'animation d'un objet puisse être le plus simple possible. Le programme est bâti de telle manière que vous pouvez parfaitement réutiliser certaines de ses routines dans vos programme personnels. Notre programme a également pour but de montrer tout ce qui peut être réalisé dans ce domaine en BASIC.

Après le lancement du programme, le menu principal apparaît sur l'écran :

```
********************************************
**                                        **
**         CPC's WORLD/TAV 8.9.86         **
**                                        **
********************************************

               .
          0       Générateur d'objet
          1       Editeur d'angles
          2       Editeur de liaisons
          3       Editeur d'animation
          4       Animation
          5       Sauver objet
          6       Charger objet
          7       Effacer objet
          8       Charger démo
          9       Fin du programme

          Choisissez une option
```

Vous pouvez sauter à l'un des dix points du menu qui vous sont proposés à l'aide d'une des touches de chiffres 0 à 9. A la fin de chaque routine, retour automatique au menu principal.

Nous allons maintenant décrire les possibilités ainsi que le mode d'emploi de chaque point du menu. Vous trouverez une description complète du programme à la suite du listing qui suit cette description.

Le générateur d'objet

Le générateur d'objet permet de définir de la façon la plus simple qui soit un modèle d'armature complexe. Après l'effacement du menu, apparaît sur l'écran un système de coordonnées en deux dimensions ainsi qu'une croix. La croix peut être déplacée dans tous les sens avec les touches curseur.

Lorsque vous actionnez la touche espace, vous faites stocker les coordonnées actuelles de la croix qui est alors marquée par la lettre "X".

Une ligne est alors automatiquement tracée de ce point à la dernière position marquée avec la touche espace. Cette méthode permet de définir la moitié bidimensionnelle d'un corps tridimensionnel. La figure suivante illustre ce mode de définition à l'aide de lignes épaisses. Les étoiles "*" indiquent où on a appuyé sur la touche espace.

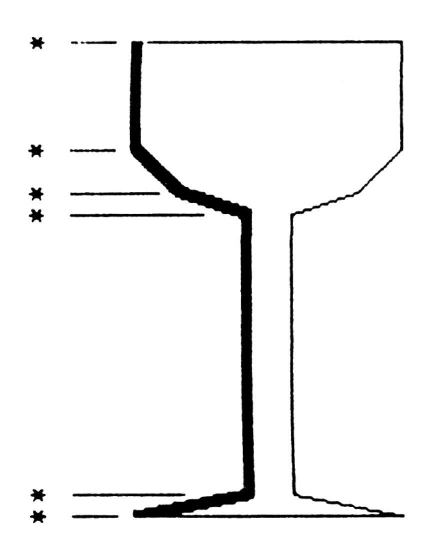


Figure 13 : **Définition d'un calice**

Pour marquer la fin de la création de l'objet, vous actionnez la touche RETURN. Il sera alors reproduit par symétrie par rapport à l'axe des Y et complété automatiquement au cours de l'étape suivante du travail.

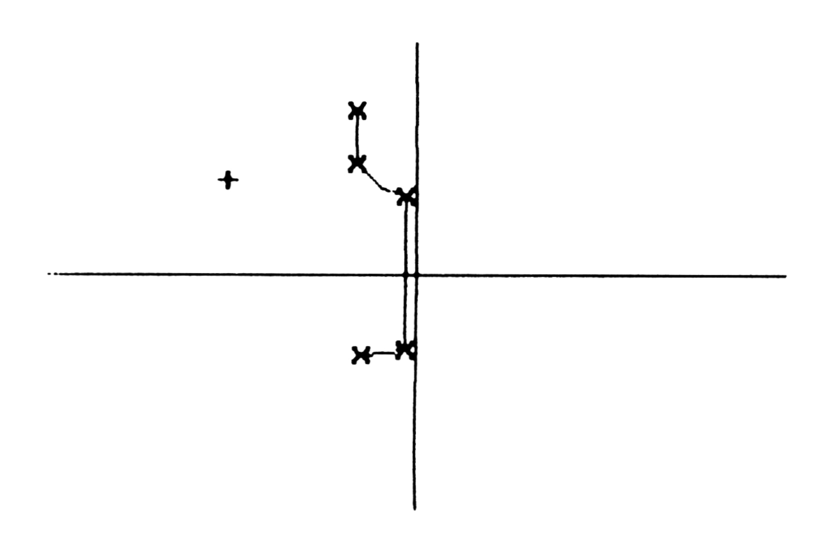


Figure 14 : **Générateur d'objet**

Editeur de sommets d'angles

Les objets peuvent également être définis à l'aide de l'éditeur de sommets d'angles (ou éditeur d'angles pour simplifier). Après que vous ayez choisi le point "1" du menu, la question

A partir du sommet?

apparaît sur l'écran. Le programme attend ici que vous entriez un nombre entre 0 et le numéro du premier sommet n'ayant pas encore été défini. Si le numéro que vous entrez est trop grand, cela vous est indiqué par le message d'erreur

*** *Ce sommet n'existe pas!* ***

Si vous actionnez la touche RETURN sans autre entrée auparavant, la routine saute toujours au sommet numéro zéro. Après la définition du premier sommet à traiter, une table de sommets apparaît sur l'écran.

SOMMET	X	Y	Z	
0	-85	94	0	Changer?
1	-66	64	0	Changer?
2	-66	30	0	Changer?
3	-28	30	0	Changer?
4	-28	-39	0	Changer?
5	-62	-39	0	Changer?

Table 2 : **Exemple de table de sommets**

Sur le côté gauche de la table figure le numéro de sommet actuel. Vous voyez affiché à côté de ce numéro le contenu de ce sommet qui est défini par des coordonnées X, Y et Z. La routine vous demande ensuite avec le message "Changer?" si la position du sommet doit être modifiée. Le programme accepte comme paramètres en entrée les réponses suivantes :

- **"O" (Oui)** qui confirme que vous voulez changer la position et fait sauter le programme à une routine d'édition.

- **"Q" (Quitter)** interrompt la fonction et revient au menu principal.

- **RETURN** ou une autre touche vous permet de passer à l'entrée suivante du tableau.

L'éditeur d'angles vous permet de définir un maximum de 128 sommets d'angles.

Editeur de liaisons

Cette routine vous permet d'établir les liaisons entre les points définis à l'aide de l'éditeur d'angles. Après que vous ayez choisi le point "2" du menu, la question

A partir de la liaison?

apparaît sur l'écran. Le programme attend ici un nombre entre 0 et le numéro de la première liaison n'ayant pas été encore définie.

Si le numéro que vous entrez est trop élevé, cela vous est indiqué par le message d'erreur

**** Cette liaison n'existe pas! ****

Si vous appuyez directement sur la touche RETURN sans entrée préalable, la routine sautera toujours à la liaison numéro zéro. Après que vous ayez défini la première liaison à traiter, une table apparaît sur l'écran.

LIAISON	P1	P2	
0	0	6	Changer?
1	6	12	Changer?
2	12	18	Changer?
3	18	0	Changer?
4	1	7	Changer?
5	7	13	Changer?

Table 3 : Exemple de table de lignes de liaison

Sur le côté gauche de la table figure le numéro de liaison actuel. Vous voyez affiché à côté de ce numéro le contenu de cette liaison qui est défini par des numéros de sommets. La routine vous demande ensuite avec le message "Changer?" si la liaison doit être modifiée. Le programme accepte les mêmes réponses que l'éditeur de sommets d'angles décrit plus haut. L'éditeur de liaisons vous permet de définir un maximum de 128 liaisons.

Editeur d'animation

La séquence de mouvements d'un objet déjà pleinement défini doit être définie avec l'éditeur d'animation. Il vous permet de définir 128 étapes d'animation, chaque étape se composant de 6 sous-étapes. L'éditeur d'animation est appelé à travers le point "3" du menu principal. Immédiatement après apparaît sur l'écran la question

A partir de la ligne?

Le programme attend ici que vous entriez un nombre entre 0 et le numéro de la première ligne de programme n'ayant pas encore été définie. Si le numéro que vous entrez est trop grand, cela vous est indiqué par le message d'erreur suivant :

*** *Cette ligne n'existe pas!* ***

Si vous actionnez la touche RETURN sans autre entrée auparavant, la routine saute toujours à la ligne numéro zéro. Après indication d'un numéro de ligne correct, une table d'animation apparaît sur l'écran.

LIGNE	XROT	YROT	ZROT	XPOS	YPOS	ZPOS	
0:	0	0	0	0	0	0	Changer?
1:	1	1	0	0	0	15	Changer?
2:	2	2	0	0	0	30	Changer?
3:	3	3	0	0	0	45	Changer?
4:	4	4	0	0	0	60	Changer?
5:	5	5	0	0	0	75	Changer?
6:	6	6	0	0	0	90	Changer?
7:	7	7	0	0	0	105	Changer?
8:	8	8	0	0	0	120	Changer?
9:	9	9	0	0	0	135	Changer?
10:	10	10	0	0	0	150	Changer?

Table 4 : Exemple de table d'animation

Sur le côté gauche de la table figure le numéro de ligne actuel. Vous voyez affiché à côté de ce numéro le contenu de cette ligne qui est défini chaque fois par trois angles de rotation et des paramètres de déplacement. La routine vous demande ensuite avec le message "Changer?" si le contenu de la ligne doit être modifié. Le programme accepte comme paramètres en entrée les réponses suivantes :

- **"O" (Oui)** qui confirme que vous voulez changer le contenu et fait sauter le programme à une routine d'édition.

- **"Q" (Quitter)** interrompt la fonction et revient au menu principal.

- **RETURN** ou une autre touche vous permet de passer à l'entrée suivante du tableau.

Si vous voulez modifier la ligne d'animation en actionnant la touche "O", vous devez d'abord définir les angles de rotation XROT, YROT et ZROT. Tout angle compris entre 0 et 360 degrés permet de décrire la rotation de l'objet sur l'axe des X, des Y ou des Z. La routine attend, pour chaque angle, une valeur positive entre 0 et 23, comme paramètre en entrée. Cette valeur représente une constante qui indique l'angle de rotation par pas de 15 degrés. La table de conversion suivante vous permet de calculer sans difficulté la constante dont vous avez besoin.

Angle	Constante	Angle	Constante
0	0	195	13
15	1	210	14
30	2	225	15
45	3	240	16
60	4	255	17
75	5	270	18
90	6	285	19
105	7	300	20
120	8	315	21
135	9	330	22
150	10	345	23
165	11	360	0
180	12		

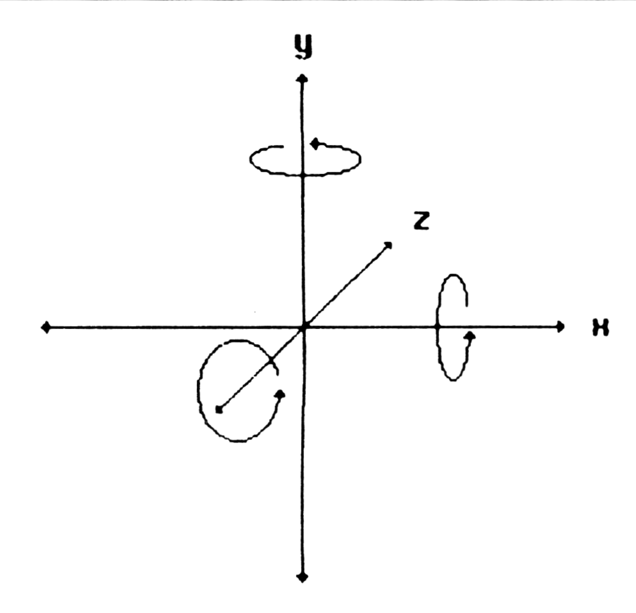


Figure 15 : **Effet de l'angle de rotation dans le système de coordonnées**

L'éditeur vous demande ensuite d'entrer les paramètres de déplacement XPOS, YPOS et ZPOS. Ils indiquent l'éloignement de l'objet sur les axes des X, des Y et des Z par rapport au point zéro. Ces paramètres peuvent être des valeurs positives ou négatives.

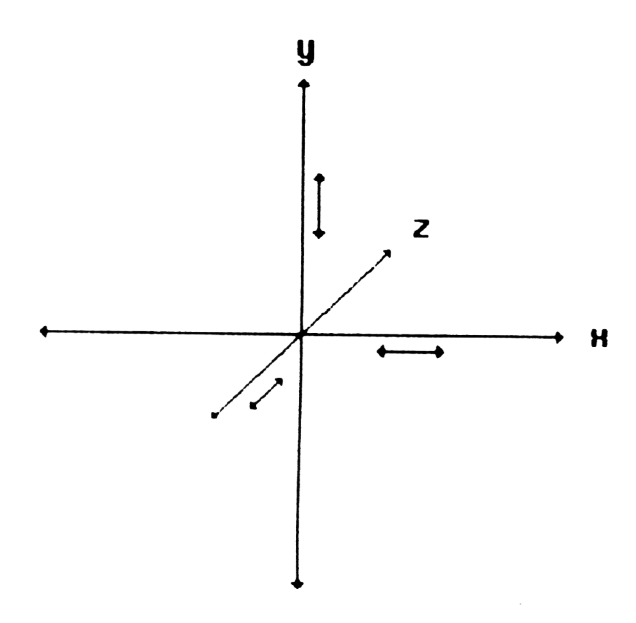


Figure 16 : **Effet des paramètres de déplacement dans le système de coordonnées**

Une fois que l'animation a été définie de cette manière, elle peut être exécutée par la fonction suivante.

Animation

Cette routine anime un objet qui a été défini d'après les données fournies par l'éditeur d'animation. Après que la dernière séquence d'animation ait été traitée, retour automatique au menu principal.

Sauvegarde de l'objet

Ce point du menu permet de sauvegarder durablement un objet. Vous pouvez en effet écrire sur la disquette tous les sommets et toutes les liaisons ainsi que la séquence d'animation que vous aurez réalisée. Après que vous ayez appelé "Sauver objet", le message

Entrez le nom de fichier?

apparaît sur l'écran.

Après entrée du nom de fichier, la sauvegarde de l'objet est effectuée avec ensuite retour automatique au menu principal.

Charger l'objet

Cette routine permet de charger un objet qui a été stocké avec la fonction "Sauver objet". Après appel de "Charger objet", le message

Entrez le nom de fichier?

apparaît sur l'écran. Après entrée du nom de fichier, le chargement de l'objet est effectué avec ensuite retour automatique au menu principal.

Effacer l'objet

Cette fonction fixe les variables des tables de sommets, de liaisons et d'animation ainsi que tous les pointeurs sur les valeurs par défaut.

Charger la démo

Charge un objet ainsi qu'une animation simple dans la mémoire.

Fin du programme

Arrête "CPCs World" avec une instruction END.

```
100 '********************************
110 '**                            **
120 '**      CPCs  WORLD           **
130 '**      8.9.'86 TAV           **
140 '**                            **
150 '********************************
160 '
170 '
180 'INITIALISATION
190 '
200 OPENOUT "DUMMY" : MEMORY &3FF3 : CLOSEOUT
210 DEG
220 DEFINT A-Z
```

```
230 DIM X(127),Y(127),Z(127),V(127,1),SI!(23),CO!(23),ALPHA(127),BE
TA(127),GAMMA(127),XOFF(127),YOFF(127),ZOFF(127)
240 FP=640
250 '
260 'CHARGER LA ROUTINE DE DEPLACEMENT DE BLOC DANS LA MEMOIRE
270 '
280 FOR I=0 TO 11:READ J:POKE &3FF4+I,J:NEXT
290 '
300 DATA &21,&00,&C0:     'LD HL,&C000
310 DATA &11,&00,&40:     'LD DE,&4000
320 DATA &01,&00,&40:     'LD BC,&4000
330 DATA &ED,&B0:         'LDIR
340 DATA &C9:             'RET
350 '
360 'PRODUIRE LES TABLES SIN ET COS
370 '
380 FOR I=0 TO 345 STEP 15:SI!(I/15)=SIN(I):CO!(I/15)=COS(I):NEXT
390 '
400 'MENU PRINCIPAL
410 '
420 MODE 1
430 PRINT"****************************************";
440 PRINT"**                                    **
450 PRINT"**";:PEN 3:PRINT"   CPCs WORLD/TAV 8.9.86         ";:P
EN 1:PRINT"**";
460 PRINT"**                                    **";
470 PRINT"****************************************";
480 PRINT:PRINT:PRINT:PRINT:PRINT
490 PEN 3:PRINT TAB(10)"0 ";:PEN 1:PRINT"Generateur d'objet"
500 PEN 3:PRINT TAB(10)"1 ";:PEN 1:PRINT"Editeur d'angles"
510 PEN 3:PRINT TAB(10)"2 ";:PEN 1:PRINT"Editeur de liaisons"
520 PEN 3:PRINT TAB(10)"3 ";:PEN 1:PRINT"Editeur d'animation"
530 PEN 3:PRINT TAB(10)"4 ";:PEN 1:PRINT"Animation"
540 PEN 3:PRINT TAB(10)"5 ";:PEN 1:PRINT"Sauver objet"
550 PEN 3:PRINT TAB(10)"6 ";:PEN 1:PRINT"Charger objet"
560 PEN 3:PRINT TAB(10)"7 ";:PEN 1:PRINT"Effacer objet"
570 PEN 3:PRINT TAB(10)"8 ";:PEN 1:PRINT"Charger demo"
580 PEN 3:PRINT TAB(10)"9 ";:PEN 1:PRINT"Fin du programme"
590 PEN 3:PRINT:PRINT:PRINT:PRINT TAB(4)"Choisissez une
option":PEN 1
600 Y$=INKEY$
610 IF Y$<"0" OR Y$>"9" THEN 600
```

```
620 CLS:ON VAL(Y$)+1 GOTO
660,1310,1420,1200,1670,990,870,1110,1530,2680
630 '
640 'CREER L'OBJET
650 '
660 GOSUB 1120:MOVE 0,200:DRAW 639,200,3:MOVE 320,0:DRAW 320,399,3
670 ORIGIN 320,200:K=0:I=-160:J=100:PRINT CHR$(23);CHR$(1) ;
680 TAG
690 GRAPHICS PEN 3:MOVE I,J:PRINT "+";
700 Y$=INKEY$:IF Y$="" THEN 700
710 MOVE I,J:PRINT "+";
720 IF Y$=CHR$(240) THEN J=J+1
730 IF Y$=CHR$(241) THEN J=J-1
740 IF Y$=CHR$(242) THEN I=I-1
750 IF Y$=CHR$(243) THEN I=I+1
760 IF Y$=" " THEN X(K)=I+8:Y(K)=J-6:K=K+1:MOVE I,J:GRAPHICS PEN
2:PRINT "X";:IF K>1 THEN MOVE X(K-2),Y(K-2):DRAW X( K-1),Y(K-1),1
770 IF Y$=CHR$(13) THEN 790
780 GOTO 690
790 FOR I=0 TO K-1:X(2*K+I)=X(I)*-1:Y(2*K+I)=Y(I):Z(K+I)=X
(I):Y(K+I)=Y(I):Z(3*K+I)=X(I)*-1:Y(3*K+I)=Y(I):NEXT
800 AK=K*4-1
810 X0=0:FOR I=0 TO AK STEP 4:FOR J=0 TO
3:V(I+J,0)=J*K+X0:V(I+J,1)=(J+1)*K+X0:NEXT:V(I+J-
1,1)=X0:X0=X0+1:NEXT
820 VB=AK:FOR J=0 TO AK STEP K:FOR I=0 TO K-
2:VB=VB+1:V(VB,0)=I+J:V(VB,1 )=I+J+1:NEXT:NEXT
830 AK=AK+1:VB=VB+1:TAGOFF:PRINT CHR$(23);CHR$(0);:GOTO 420
840 '
850 'CHARGER L'OBJET
860 '
870 PEN 3:PRINT TAB(10)"*** CHARGER L'OBJET ***":PEN 1
880 LOCATE 1,12:INPUT"Entrez le nom de fichier";Y$
890 GOSUB 1120:OPENIN Y$
900 INPUT #9,AA,AK,VB
910 FOR I=0 TO AA:INPUT #9,ALPHA(I),BETA(I),GAMMA(I),XOFF(I),YOFF(I
),ZOFF(I):NEXT
920 FOR I=0 TO AK:INPUT #9,X(I),Y(I),Z(I):NEXT
930 FOR I=0 TO VB:INPUT #9,V(I,0),V(I,1):NEXT
940 CLOSEIN
950 GOTO 420
960 '
```

```
970 'STOCKER L'OBJET
980 '
990 PEN 3:PRINT TAB(10)"*** SAUVER OBJET ***":PEN 1
1000 LOCATE 1,12:INPUT"Entrez le nom de fichier";Y$
1010 OPENOUT Y$
1020 WRITE #9,AA,AK,VB
1030 FOR I=0 TO AA:WRITE #9,ALPHA(I),BETA(I),GAMMA(I),XOFF(I),YOFF(
I),ZOFF(I):NEXT
1040 FOR I=0 TO AK:WRITE #9,X(I),Y(I),Z(I):NEXT
1050 FOR I=0 TO VB:WRITE #9,V(I,0),V(I,1):NEXT
1060 CLOSEOUT
1070 GOTO 420
1080 '
1090 EFFACER L'OBJET
1100 '
1110 GOSUB 1120:GOTO 420
1120 AA=0:AK=0:VB=0
1130 FOR I=0 TO 127
1140 ALPHA(I)=0:BETA(I)=0:GAMMA(I)=0:XOFF(I)=0:YOFF(I)=0:ZOFF(I)=0:
V(I,0)=0:V(I,1)=0:X(I)=0:Y(I)=0:Z(I)=0
1150 NEXT
1160 RETURN
1170 '
1180 'EDITEUR D'ANIMATION
1190 '
1200 INPUT"A partir de la ligne";I:IF I>AA OR I<0 THEN PRINT"***
Cette ligne n'existe pas! ***":PRINT:GOTO 1200
1210 MODE 2:PRINT"LIGNE    XROT    YROT    ZROT    XPOS    YPOS
ZPOS"
1220 PRINT"  ";USING"###";I;:PRINT":       ";USING"##";ALPHA(I);:PRIN
T"       ";USING"##";BETA(I);:PRINT"       ";USING"##";GAMMA(I);:PRIN
T"     ";USING"####";XOFF(I);:PRINT"     ";USING"####";YOFF(I);:PRINT
"     ";USING"####";ZOFF(I);
1230 INPUT"     Changer ligne";Y$:Y$=UPPER$(Y$)
1240 IF Y$<>"O" AND I+1>AA OR Y$="Q" THEN 420 ELSE IF Y$<>"O" THEN
1260
1250 PRINT TAB(9);:INPUT ALPHA(I):PRINT CHR$(11);STRING$(16,9);:INP
UT BETA(I):PRINT CHR$(11);STRING$(24,9);:INPUT GAMMA(I):PRINT CHR$(
11);STRING$(32,9);:INPUT XOFF(I):PRINT CHR$(11);STRING$(40,9);:INPU
T YOFF(I):PRINT CHR$(11);STRING$(48,9);:INPUT ZOFF(I)
1260 I=I+1:IF I>AA THEN AA=I
1270 IF I>127 THEN AA=127:GOTO 420 ELSE 1220
```

```
1280 '
1290 'EDITEUR DE SOMMETS
1300 '
1310 INPUT"A partir du sommet";I:IF I>AK OR I<0 THEN PRINT"*** Ce
sommet n'existe pas! ***":PRINT:GOTO 1310
1320 CLS:PEN 3:PRINT"SOMMET      X       Y       Z":PEN 1
1330 PRINT"   ";USING"###";I;:PRINT"      ";USING"####";X(I);:PRINT"
";USING"####";Y(I);:PRINT"    ";USING"####";Z(I);
1340 INPUT"   Changer";Y$:Y$=UPPER$(Y$)
1350 IF Y$<>"O" AND I+1>AK OR Y$="Q" THEN 420 ELSE IF Y$<>"O" THEN
1370
1360 PRINT TAB(9);:INPUT X(I):PRINT CHR$(11);STRING$(15,9);:INPUT Y
(I):PRINT CHR$(11);STRING$(22,9);:INPUT Z(I)
1370 I=I+1:IF I>AK THEN AK=I
1380 IF I>127 THEN AK=127:GOTO 420 ELSE 1330
1390 '
1400 'EDITEUR DE LIAISONS
1410 '
1420 INPUT"A partir de la liaison";I:IF I>VB OR I<0 THEN PRINT"***
Cette liaison n'existe pas! ***":PRINT:GOTO 1420
1430 CLS:PEN 3:PRINT"LIAISON       P1       P2":PEN 1
1440 PRINT"   ";USING"###";I;:PRINT"            ";USING"###";V(I,0);:PRI
NT"      ";USING"###";V(I,1);
1450 INPUT"   Changer";Y$:Y$=UPPER$(Y$)
1460 IF Y$<>"O" AND I+1>VB OR Y$="Q" THEN 420 ELSE IF Y$<>"O" THEN
1480
1470 PRINT TAB(13);:INPUT V(I,0):PRINT CHR$(11);STRING$(20,9);:INPU
T V(I,1)
1480 I=I+1:IF I>VB THEN VB=I
1490 IF I>127 THEN VB=127:GOTO 420 ELSE 1440
1500 '
1510 'CHARGER LA DEMO
1520 '
1530 GOSUB 1120:RESTORE 2120
1540 READ AK
1550 FOR I=0 TO AK-1:READ X(I),Y(I),Z(I):NEXT
1560 '
1570 READ VB
1580 FOR I=0 TO VB-1:READ V(I,0),V(I,1):NEXT
1590 '
1600 READ AA
1610 FOR I=0 TO AA-1:READ
```

```
ALPHA(I),BETA(I),GAMMA(I),XOFF(I),YOFF(I),ZOFF (I):NEXT
1620 '
1630 GOTO 420
1640 '
1650 'BOUCLE D'ANIMATION
1660 '
1670 ORIGIN 320,200
1680 FOR K=0 TO AA-1
1690 '
1700 'DISPLAY MANAGEMENT
1710 '
1720 OUT &BC00,12:OUT &BD00,52
1730 CALL &3FF4
1740 OUT &BC00,12:OUT &BD00,16
1750 CLS
1760 '
1770 P0!=CO!(GAMMA(K))*CO!(BETA(K))
1780 P1!=SI!(GAMMA(K))*CO!(BETA(K))
1790 P2!=-SI!(BETA(K))
1800 P3!=-
SI!(GAMMA(K))*CO!(ALPHA(K))+CO!(GAMMA(K))*SI!(BETA(K))*SI!(ALP
HA(K))
1810 P4!=CO!(GAMMA(K))*CO!(ALPHA(K))+SI!(GAMMA(K))*SI!(BETA(K))*SI!
(ALPHA(K))
1820 P5!=CO!(BETA(K))*SI!(ALPHA(K))
1830 P6!=SI!(GAMMA(K))*SI!(ALPHA(K))+CO!(GAMMA(K))*SI!(BETA(K))*CO!
(ALPHA(K))
1840 P7!=-
CO!(GAMMA(K))*SI!(ALPHA(K))+SI!(GAMMA(K))*SI!(BETA(K))*CO!(ALP
HA(K))
1850 P8!=CO!(BETA(K))*CO!(ALPHA(K))
1860 '
1870 'BOUCLE DE PROJECTION
1880 '
1890 FOR I=0 TO VB-1
1900 J=0:GOSUB 2040
1910 MOVE XP,YP
1920 J=1:GOSUB 2040
1930 DRAW XP,YP
1940 NEXT
1950 '
1960 NEXT
```

```
1970 '
1980 OUT &BC00,12:OUT &BD00,52
1990 FOR I=1 TO 10000:NEXT
2000 GOTO 420
2010 '
2020 'CONVERSION 3 D
2030 '
2040 X0=X(V(I,J)):Y0=Y(V(I,J)):Z0=Z(V(I,J))
2050 ZP!=1-(X0*P2!+Y0*P5!+Z0*P8!+ZOFF(K))/FP
2060 XP=(X0*P0!+Y0*P3!+Z0*P6!+XOFF(K))/ZP!
2070 YP=(X0*P1!+Y0*P4!+Z0*P7!+YOFF(K))/ZP!
2080 RETURN
2090 '
2100 'COORDONNEES
2110 '
2120 DATA 8
2130 DATA -50,-50,-50
2140 DATA -50,50,-50
2150 DATA 50,50,-50
2160 DATA 50,-50,-50
2170 DATA -50,-50,50
2180 DATA -50,50,50
2190 DATA 50,50,50
2200 DATA 50,-50,50
2210 '
2220 'LIAISONS
2230 '
2240 DATA 14
2250 DATA 0,1
2260 DATA 1,2
2270 DATA 2,3
2280 DATA 3,0
2290 DATA 4,5
2300 DATA 5,6
2310 DATA 6,7
2320 DATA 7,4
2330 DATA 0,4
2340 DATA 1,5
2350 DATA 2,6
2360 DATA 3,7
2370 DATA 0,2
2380 DATA 1,3
```

```
2390 '
2400 ANIMATION
2410 '
2420 DATA 25
2430 DATA 0,0,0,0,0,0
2440 DATA 1,0,0,0,0,0
2450 DATA 2,0,0,0,0,0
2460 DATA 3,0,0,0,0,0
2470 DATA 4,0,0,0,0,0
2480 DATA 5,0,0,0,0,0
2490 DATA 6,0,0,0,0,0
2500 DATA 7,0,0,0,0,0
2510 DATA 8,0,0,0,0,0
2520 DATA 9,0,0,0,0,0
2530 DATA 10,0,0,0,0,0
2540 DATA 11,0,0,0,0,0
2550 DATA 12,0,0,0,0,0
2560 DATA 13,0,0,0,0,0
2570 DATA 14,0,0,0,0,0
2580 DATA 15,0,0,0,0,0
2590 DATA 16,0,0,0,0,0
2600 DATA 17,0,0,0,0,0
2610 DATA 18,0,0,0,0,0
2620 DATA 19,0,0,0,0,0
2630 DATA 20,0,0,0,0,0
2640 DATA 21,0,0,0,0,0
2650 DATA 22,0,0,0,0,0
2660 DATA 23,0,0,0,0,0
2670 DATA 0,0,0,0,0,0
2680 END
```

Description du programme :

200-240 *Bloc d'initialisation*
La mémoire est protégée, le calcul en degrés est désactivé et tous les vecteurs et matrices sont dimensionnés. Il faut noter que la protection de la mémoire se fait alors qu'un fichier DUMMY est ouvert pour que le système d'exploitation puisse réserver suffisamment de place pour le buffer du lecteur. Toutes les variables sont par ailleurs déclarées comme variables entières (DEFINT) et la variable FP (point de fuite) est fixée sur sa valeur par défaut.

250-340 *Charger la routine de déplacement de bloc dans la mémoire*
Les éléments DATA contenus dans les lignes BASIC 300 à 340 sont POKés dans la zone protégée de la RAM à partir de &3FF4.

350-380 *Produire les tables de sinus et cosinus*
On calcule ici les valeurs SIN et COS par intervalle de 15 degrés. La variable de comptage reste comprise entre 0 et 23 du fait d'une division par 15.

390-620 *Produire le menu principal*
Le CPC est placé en mode graphique 1 ce qui vide l'écran. Le menu principal est ensuite écrit sur l'écran avec des instructions PRINT. En lignes 600 et 610 est effectué un test du clavier suivi d'un contrôle de validité des paramètres en entrée. Si les paramètres entrés sont corrects, la ligne 620 saute à la routine appelée.

630-830 *Créer l'objet* ·
Les lignes 660 à 680 dessinent le système de coordonnées et initialisent les valeurs défaut. Les lignes 690 à 780 contiennent la boucle d'animation de la croix ainsi que le test et l'évaluation des touches du curseur et RETURN. Les lignes suivantes calculent les valeurs symétriques aux coordonnées entrées par rapport à l'axe des Y.

840-950 *Charger l'objet*
Après que toutes les tables aient été effacées par GOSUB 1120, les trois premiers octets des données définies par Y$ sont transférés dans les variables AA, AK et VB. AA contient alors le nombre des séquences d'animation, AK le nombre de sommets et VB le nombre de liaisons qui seront affectées aux différentes tables par les lignes 910 à 930.

960-1070 *Sauvegarder l'objet*
Les variables AA, AK et VB, qui sont écrites en premier dans le fichier défini par Y$, contiennent le nombre d'éléments stockés respectivement dans les tables d'animation, de sommets et de liaisons. C'est pourquoi elles sont également utilisées dans les lignes 1030 à 1050 comme variables de comptage dans les boucles FOR-TO-NEXT qui écrivent sur disquette le contenu des différentes tables.

1080-1160 *Effacer l'objet*
L'instruction GOSUB de la ligne 1110 appelle la routine de suppression et termine la fonction par un retour au menu principal (GOTO 420). Comme la fonction de suppression a été définie sous forme d'un sous-programme, elle peut être appelée à partir d'autres modules du programme.

Les lignes 1120 à 1160 contiennent la véritable routine de suppression. Les variables AA, AK et VB y sont fixées sur zéro et toutes les valeurs des tables sont effacées par la boucle FOR-TO-NEXT des lignes 1130 à 1150.

1170-1270 *Editeur d'animation*
Le numéro de la première ligne d'animation est défini en ligne 1200 comme variable de comptage I. Le contenu de cette ligne est ensuite sorti sur l'écran, par des instructions USING, sous la forme des variables ALPHA(I), BETA(I), GAMMA(I), XOFF(I), YOFF(I) et ZOFF(I).

Les lignes 1230 à 1240 procèdent ensuite à l'entrée du paramètre EDIT dans Y$, sortent le message "Changer ligne?" et vérifient enfin la validité de l'entrée.

- Si Y$ contient "O", la ligne 1250 effectue, avec des instructions INPUT, une entrée formatée des variables ALPHA(I), BETA(I), GAMMA(I), XOFF(I), YOFF(I) et ZOFF(I).

- Si le paramètre "Q" est identifié, retour au menu principal. Si Y$ ne contient pas "O" et si I+1 est supérieur au contenu de la variable AA (lignes maxi), il y a également retour au menu principal.

- Si Y$ contient n'importe quel autre caractère, saut à la ligne 1260.

Les lignes 1260 à 1270 incrémentent la variable de comptage I et vérifient que son contenu ne dépasse pas la valeur 127. Ensuite la variable AA (lignes maxi) est adaptée à la variable de comptage I si nécessaire.

1280-1380 *Editeur d'angles*

Après que le numéro du premier sommet d'angles ait été défini en ligne 1310 comme variable de comptage I, son contenu est sorti sur l'écran, par des instructions USING, sous la forme des variables X(I), Y(I) et Z(I). Le paramètre EDIT est alors entré dans Y$, dans les lignes 1340 à 1350, qui sortent ensuite le message "Changer?" et contrôlent la validité de l'entrée.

- Si Y$ contient "O", la ligne 1360 effectue, avec des instructions INPUT, une entrée formatée des variables X(I), Y(I), et Z(I).

- Si le paramètre "Q" est identifié, retour au menu principal. Si Y$ ne contient pas "O" et si I+1 est supérieur au contenu de la variable AK (lignes maxi), il y a également retour au menu principal.

- Si Y$ contient n'importe quel autre caractère, saut à la ligne 1370.

Les lignes 1370 à 1380 incrémentent la variable de comptage I et vérifient que son contenu ne dépasse pas la valeur 127. Ensuite la variable AK (lignes maxi) est adaptée à la variable de comptage I si nécessaire.

1390-1490 *Editeur de liaisons*
Le numéro de la première ligne d'animation est défini en ligne 1420 comme variable de comptage I. Le contenu de cette ligne est ensuite sorti sur l'écran, par des instructions USING, sous la forme des variables V(I,0) et V(I,1). Les lignes 1450 à 1460 procèdent ensuite à l'entrée du paramètre EDIT dans Y$, sortent le message "Changer ligne?" et vérifient enfin la validité de l'entrée.

- Si Y$ contient "O", la ligne 1470 effectue, avec des instructions INPUT, une entrée formatée des variables V(I,0) et V(I,1).

- Si le paramètre "Q" est identifié, retour au menu principal. Si Y$ ne contient pas "O" et si I+1 est supérieur au contenu de la variable VB (lignes maxi), il y a également retour au menu principal.

- Si Y$ contient n'importe quel autre caractère, saut à la ligne 1480.

Les lignes 1480 à 1490 incrémentent la variable de comptage I et vérifient que son contenu ne dépasse pas la valeur 127.

Ensuite la variable VB (lignes maxi) est adaptée à la variable de comptage I si nécessaire.

1500-1630 *Charger la démo*
Après que toutes les tables aient été effacées (GOSUB 1120) et que le pointeur DATA ait été positionné sur le début des blocs de données de démonstration, les lignes 1540 à 1610 chargent les éléments DATA dans les tables de sommets, de liaisons et d'animation. Les instructions READ placées avant chaque boucle chargent dans la variable correspondante pour le nombre maxi de lignes (AA, AK ou VB) le nombre d'éléments de données + 1. Ce nombre sera réutilisé dans la boucle FOR-TO-NEXT placée à la ligne suivante qui se chargera de lire ces données.

1640-1680 *Boucle d'animation*
La boucle FOR-TO-NEXT en ligne 1680 est chargée de la commande des séquences d'animation. La variable de comptage K représente à cet égard la clé d'accès à l'animation (l'index). L'instruction ORIGIN 320,200 de la ligne 1670 fixe le système de coordonnées dans le centre de l'écran.

1690-1750 *Display Management*
Cette routine est chargée de la gestion des deux pages d'écran. Cette méthode permet de réaliser le dessin sur une page d'écran invisible pour éviter un tremblotement désagréable de l'image. La ligne 1720 active la page d'écran normale (&C000 à &FFFF). Le contenu de l'écran est ensuite copié dans BANK 1 (&4000 à &7FFF). L'instruction OUT de la ligne 1740 place la mémoire écran en &4000 et rend ainsi visible le contenu de BANK 1. La page d'écran normale (&C000 à &FFFF) peut alors être vidée avec l'instruction CLS.

1760-1850 *Définition des paramètres de rotation*
Ces lignes calculent à l'aide de tables trigonométriques les paramètres P0 à P8 nécessaires pour la rotation.

1860-2000	*Boucle de projection* Cette boucle va chercher toutes les lignes d'un objet dans les tables et retransmet les indices des coordonnées de départ et de fin à la routine de conversion 3 D (GOSUB 2040). Les coordonnées X1, X2, Y1 et Y2 obtenues en XP et YP sont utilisées pour le dessin des lignes dans les lignes de programme 1910 et 1930. Une fois la séquence d'animation terminée, les lignes 1980 à 1990 font que la dernière image placée dans le buffer écran peut rester quelques secondes sur l'écran pour que l'utilisateur puisse l'examiner.
2010-2080	*Conversion 3 D* Cette routine calcule les positions X et Y d'un sommet sur l'écran, en perspective centrale, d'après les paramètres de rotation P0 à P8 et d'après la variable de point de fuite FP.
2090-2200	*Sommets pour la démo* Le premier élément DATA indique le nombre de lignes DATA suivantes.
2210-2380	*Liaisons pour la démo* Le premier élément DATA indique le nombre de lignes DATA suivantes.
2390-2670	*Séquences d'animation pour la démo* Le premier élément DATA indique le nombre de lignes DATA suivantes.
2680	*Fin du programme* Cette instruction END ne doit pas être supprimée car elle est appelée par la fonction "fin du programme".

7.5.1 Amélioration du programme

Nous souhaitons ici vous donner quelques idées de modules d'extension du programme 3 D. La plupart de ces idées ne peuvent malheureusement être réalisées qu'en langage machine. Nous espérons cependant que les lecteurs passionnés par ces problèmes pourront améliorer "CPCs World" ou réaliser d'après nos suggestions un programme 3 D personnel.

Les lignes cachées (Hidden Lines)

Tous les algorithmes présentés jusqu'ici projetaient sur l'écran même les côtés d'un objet qui ne sont pas visibles normalement. Pour remédier à cela, il n'est pas nécessaire de modifier la méthode de projection utilisée. Il suffit d'intégrer dans le programme 3 D un module dit "hidden line". Ce module "hidden line" aura pour tâche de manipuler toutes les coordonnées de telle façon que notre routine de projection ne dessine pas les "lignes cachées".

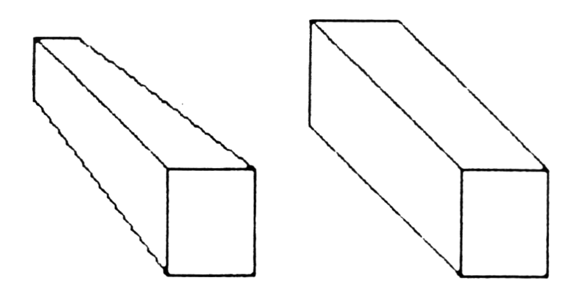


Figure 17 : **Objet avec lignes cachées : à droite perspective parallèle, à gauche perspective centrale**

Un tel algorithme de lignes cachées peut être réalisé de diverses façons. On peut par exemple doter d'un index toutes les surfaces d'un objet triées par rapport à l'axe des Z. Ces surfaces seront alors placées dans une table.

Cette table permettra ensuite de décider quelles coordonnées doivent être retransmises à la routine de projection et lesquelles doivent être ignorées.

Objets multicolores

Il y a plusieurs possibilités pour intégrer la richesse de couleurs du CPC AMSTRAD dans des programmes 3 D. Si vous animez simultanément plusieurs objets sur l'écran, vous pouvez par exemple doter chaque armature d'une couleur différente. Si l'animation se limite à un seul objet, vous pouvez modifier l'éditeur de liaisons pour qu'il vous permette de définir pour chaque ligne un paramètre de couleur en plus des paramètres de liaison. Il est naturellement tout à fait possible de combiner ces deux procédés.

Le coloriage le plus intéressant consiste certainement à colorer dans différentes couleurs toutes les surfaces d'un objet. Une telle opération suppose cependant que vous disposiez d'une fonction Fill (remplissage de surfaces) ainsi que d'un algorithme de lignes cachées.

Ombre et lumière

La définition d'une source de lumière imaginaire dans l'espace tridimensionnel constitue un thème particulièrement fascinant. A cet effet, il faut que les surfaces de tous les objets projetés soient remplies de couleurs différentes. Les surfaces placées directement sous l'effet de la lumière devront être remplies avec une couleur claire. La meilleure façon de simuler la pénombre est d'employer une couleur sombre. L'ombre complète sera représentée en noir et les sources de lumière en blanc brillant.

Il existe cependant une autre possibilité d'obtenir de tels effets qui est en même temps assez simple. Au lieu du calcul complexe des ombres, pénombres, sources de lumières diffuses et des luminaires, il suffit que la clarté des couleurs respecte un certain rapport à l'axe des Z. Cela signifie que la surface ayant la valeur Z la plus élevée aura aussi la couleur la plus claire et que la surface avec la valeur Z la plus faible recevra au contraire la couleur la plus sombre.

Une simulation d'ombres de ce type peut améliorer très nettement l'effet tridimensionnel produit par votre dessin, bien qu'elle soit fondée sur des principes mathématiques erronés. Elle suppose cependant que vous disposiez d'une fonction Fill ainsi que d'un algorithme hidden line.

Animation sans à-coups

Une animation sans à-coups d'armatures tridimensionnelles, pour peu qu'elle soit possible sans des ordinateurs beaucoup plus puissants, ne peut en tout cas être obtenue qu'en langage machine. Le présent ouvrage ne peut certainement pas, et ce n'est pas son but, vous transmettre tout le savoir-faire nécessaire pour cela. La programmation en langage machine Z80 constitue un domaine particulier qu'il faut étudier avec la littérature spécialisée. Pour ceux qui savent déjà programmer en langage machine, nous indiquerons tout de même les principales règles pour convertir de la façon la plus efficace les algorithmes 3 D que nous vous avons présentés en BASIC.

1. N'oubliez pas que l'unité centrale Z80 est un processeur orienté registres. Il est donc toujours préférable, lorsque c'est possible, d'utiliser un registre plutôt qu'une cellule de la mémoire.

2. Réalisez toujours les routines d'exécution relativement lente en vous aidant d'un manuel du processeur Z80. Essayez de réduire au maximum la durée globale d'exécution de votre routine. Vous devez pour cela additionner les cycles d'horloge de toutes les instructions utilisées et vérifier si certaines séquences d'instructions ne peuvent pas être remplacées par d'autres instructions nécessitant moins de cycles d'horloge. Soyez avare de chaque microseconde car celles-ci se cumulent vite, surtout dans les boucles de programme!

3. Ecrivez votre propre algorithme de tracé de ligne. L'instruction DRAW du CPC est rapide mais on peut aller encore beaucoup plus vite. L'idéal serait d'ailleurs de se passer de toutes les fonctions du système d'exploitation (y compris pour la division et la multiplication).

4. Réalisez toujours toute forme de sortie écran avec des octets entiers (masques bits).

5. Utilisez autant de tables que possible. N'oubliez pas que l'idéal est qu'un élément de table ne comporte qu'un octet. Il faudrait que même les tables trigonométriques ne comportent que des données sur un octet.

Et pour terminer encore un conseil pour ceux qui trouvent que la conversion de programmes en langage machine donne trop de travail. Une animation d'objet sans à-coups peut aussi être réalisée, dans certaines limites, avec un programme BASIC et deux petites routines en langage machine. Comment? C'est tout simple : après le calcul de chaque séquence, on appelle une petite routine machine qui comprime au maximum les données de l'écran de 16K et qui écrit le résultat sur la disquette. Une fois qu'une séquence complète a été sauvegardée de cette façon, il suffit qu'un petit programme machine charge toutes les images comprimées sauvegardées dans la mémoire et les envoie l'une après l'autre dans la mémoire écran.

8. Allons plus loin - les périphériques

L'expérience montre que les ordinateurs restent rarement seuls et qu'ils entraînent au contraire en général à leur suite sur le marché tout un parc d'appareils. Cela commence par des extensions de mémoire et des interfaces supplémentaires que vous pouvez implanter directement sur votre machine et cela va jusqu'à de véritables machines périphériques. C'est justement aux périphériques pouvant apporter quelque chose dans le domaine du graphisme, pour peu qu'on sache comment les employer, que nous consacrerons ce chapitre.

8.1 L'INTERROGATION DU JOYSTICK

(manche à balai ou manette)

Les joysticks connectés sur le CPC sont interrogés par celui-ci comme une partie du clavier. Outre cette possibilité d'interrogation automatique, l'état des deux joysticks peut cependant également être déterminé à l'aide de JOY(0) et JOY(1). Ces fonctions fournissent un résultat en code bits qui reflète la direction du déplacement du joystick au moment de la dernière interrogation. Il s'agit d'un code bits parce que chacun des bits de la valeur restituée par la fonction JOY a une signification particulière. La table suivante illustre ce principe pour un joystick dont le manche est appuyé vers l'avant :

Valeur fournie par la fonction JOY 00000001

Bit 3:	droite
Bit 2:	gauche
Bit 1:	arrière
Bit 0:	avant

Il est aisé de passer de l'interrogation du joystick à son application dans la programmation graphique. C'est ainsi que le joystick permet aisément de déplacer une marque, appelons-la curseur joystick, sur l'écran. Le curseur joystick peut être utilisé dans la programmation d'un programme de dessin pour marquer l'endroit où est situé le pinceau ou le crayon à dessin.

Voici maintenant un programme qui illustre le principe de la commande d'un curseur joystick. Les déplacements du joystick sont convertis en modifications de position du curseur joystick calculées au point d'image près. Le curseur joystick est représenté par un "X" qui peut être déplacé à volonté à l'intérieur de la fenêtre d'écran. Le programme détecte et interdit les déplacements qui feraient sortir le curseur joystick de l'écran.

```
100 DEFINT A-Z
110 MODE 1
120 TAG
130 X=320
140 Y=200
150 XOLD=X
160 YOLD=Y
170 MOVE X,Y
180 PRINT "X";
190 '
200 J=JOY(0)
210 IF J=0 THEN 200
220 '
230 IF (J AND 1)=1 THEN Y=Y+1
240 IF (J AND 2)=2 THEN Y=Y-1
250 IF (J AND 4)=4 THEN X=X-1
260 IF (J AND 8)=8 THEN X=X+1
270 '
280 IF X<0 THEN X=0
290 IF X>623 THEN X=623
300 IF Y<15 THEN Y=15
310 IF Y>399 THEN Y=399
320 '
330 MOVE XOLD,YOLD
340 PRINT " ";
350 '
360 MOVE X,Y
```

```
370 PRINT "X";
380 '
390 XOLD=X
400 YOLD=Y
410 '
420 GOTO 200
```

Description du programme :

100-180 La partie d'initialisation du programme définit toutes les variables comme des variables entières, fixe le MODE 1 et lie la position de la sortie de texte aux coordonnées du curseur graphique. Les variables X et Y, qui contiennent les coordonnées graphiques de la situation du curseur joystick, sont fixées sur leur valeur par défaut (le centre de l'écran). Leurs valeurs sont affectées aux variables XOLD et YOLD dans lesquelles est conservée l'ancienne position du curseur joystick. A la fin de cette partie du programme, le curseur graphique est fixé avec les paramètres X et Y et le curseur joystick est écrit sur l'écran sous la forme d'un "X".

190-210 L'état du joystick est déterminé à l'aide de la fonction BASIC JOY(0) et il est affecté à la variable J. Si J vaut 0, c'est-à-dire si le joystick est au repos, retour à une nouvelle interrogation du joystick.

220-260 On teste chacun des bits 0 à 3 de la variable J pour savoir s'il est mis (=1) ou annulé (=0). Les coordonnées du curseur joystick sont alors actualisées en fonction du déplacement du joystick reflété par ces tests.

270-310 On teste dans ces lignes de programme si la position obtenue sera en dehors de l'écran. Si c'est le cas, on effectue la correction nécessaire.

320-Fin Les lignes de programme restantes effacent le curseur joystick dans son ancienne position sur l'écran (XOLD, YOLD) et le fixent sur les nouvelles coordonnées (X, Y). Une fois le curseur joystick placé sur l'écran, les nouvelle coordonnées deviennent les anciennes et le programme fait une boucle.

8.2 LE CRAYON OPTIQUE

Le crayon optique est un périphérique qui fait encore assez "classe" et dont on dit souvent qu'il est réservé aux systèmes professionnels. L'apparence extérieure de ce petit appareil n'est pas très différente de celle d'un crayon auquel on aurait attaché un câble le reliant à l'ordinateur. Mais si on examine de plus près la pointe du crayon, on y découvre un petit détecteur optique qui constitue l'élément le plus important du crayon optique.

Le principe de fonctionnement du crayon optique est très simple. Lorsque sa pointe, qui comporte le détecteur optique, est dirigée vers l'écran de l'ordinateur, il produit une impulsion chaque fois que le rayon cathodique frappe sa position sur l'écran. Cette impulsion peut être évaluée par l'ordinateur, avec un programme approprié, ce qui permet de déterminer la position du crayon optique sur l'écran. Ce mode de fonctionnement permet tout une série d'applications possibles du crayon optique. Ces applications vont de la simple sélection dans un menu au dessin sur l'écran.

Ce chapitre essaiera donc de vous montrer que le crayon optique n'est pas réservé aux systèmes professionnels et qu'il peut parfaitement être utilisé sur le CPC. Nous commencerons nos explications en vous indiquant comment construire vous-même un crayon optique élémentaire qui ne fonctionnera cependant qu'avec un moniteur couleur. Nous vous donnerons ensuite un programme illustrant l'emploi du crayon optique dans un exemple de sélection dans un menu.

Pour construire un crayon optique, vous n'avez besoin que de quelques composants que vous devriez trouver en principe dans n'importe quelle boutique d'électronique. Le branchement peut être construit même par les plus inexpérimentés sur une plaque perforée

que vous devriez également pouvoir trouver facilement dans une boutique d'électronique. Le photo-transistor BPX 81 ne doit pas être raccordé directement sur la carte. Il vaut mieux le raccorder au reste du circuit par un cordon de deux fils.

Le photo-transistor, dont la face sensible à la lumière devra être dirigée vers l'écran lorsqu'il fonctionnera, devrait de préférence être placé dans une enveloppe de protection car il est petit et peut aisément être endommagé. L'enveloppe plastique d'un stylo bille que vous aurez débarrassée de tout ce qu'elle contenait conviendrait parfaitement pour le BPX 81. L'orifice de la pointe du crayon devra auparavant être travaillé avec une lime ou un couteau pointu de façon à ce que le photo-transistor et les connexions que vous aurez raccordés préalablement puissent y être fixés avec de la colle. Attention lors du collage à ce que la colle ne goutte pas sur la partie photo-sensible du composant car cela affecterait considérablement son bon fonctionnement.

Une fois le circuit mis en place, une fois que le photo-transistor aura été collé, il ne vous reste plus en fait qu'à connecter le tout à votre CPC AMSTRAD. Il faut absolument utiliser pour cela une prise appropriée même si elle constituera finalement le facteur de coût le plus important pour le crayon optique. Lors du raccordement du câble entre le circuit et la prise, ne tenez pas compte des inscriptions portées sur les broches de certaines prises. L'ordinateur AMSTRAD emploie une désignation des broches qui s'écarte de la norme habituelle.

Trois connexions de la prise de 50 pôles jouent un rôle pour le crayon optique : la broche 27 avec la tension positive (5 volts), la broche 49, la masse, et la broche 47 qui est appelée crayon optique. Dès lors que nous parlons de cette connexion, nous en arrivons à la description du circuit du crayon optique.

Lorsque le circuit est en fonction, le photo-transistor est dirigé sur l'écran du CPC. Chaque fois que le rayon cathodique atteindra, lors de la construction de l'image, l'endroit où est situé le crayon optique, le photo-transistor enregistrera l'impulsion lumineuse produite par le passage du rayon cathodique. Comme l'impulsion produite par le photo-transistor est trop faible pour permettre une autre exploitation, cette impulsion est renforcée par le transistor Darlington BC 517.

La résistance ajustable de 10 K permet de régler la sensibilité du crayon optique. L'impulsion ainsi préparée est enfin transmise au circuit à seuil TTL qui dote l'impulsion d'un niveau TTL. Pour obtenir à partir de cette impulsion un front montant (low-high), l'impulsion doit être inversée par une seconde porte logique. Voilà le chemin qui conduit, à travers le circuit électronique du crayon optique, de l'impulsion déclenchée par le rayon cathodique à la broche 47 (crayon optique) du CPC.

Lorsqu'un front montant (low-high) apparaît sur la broche 47, l'état actuel des Memory-Adress-Lines est transféré et stocké dans le registre crayon optique du contrôleur vidéo. C'est à travers les Memory-Adress-Lines que sont adressées les cellules de la mémoire écran mais c'est aussi, inversement, d'après leur situation qu'on peut déterminer la position du crayon optique sur l'écran au moment de l'impulsion. La situation des Memory-Adress-Lines, telle qu'elle est stockée dans le registre crayon optique, peut ensuite être lue et évaluée par un programme.

Si vous n'êtes pas persuadé d'avoir tout saisi dans cette description des phénomènes liés au fonctionnement du crayon optique, rassurez-vous : vous pouvez parfaitement vous lancer dans la construction du crayon optique sans savoir précisément comment il est censé fonctionner.

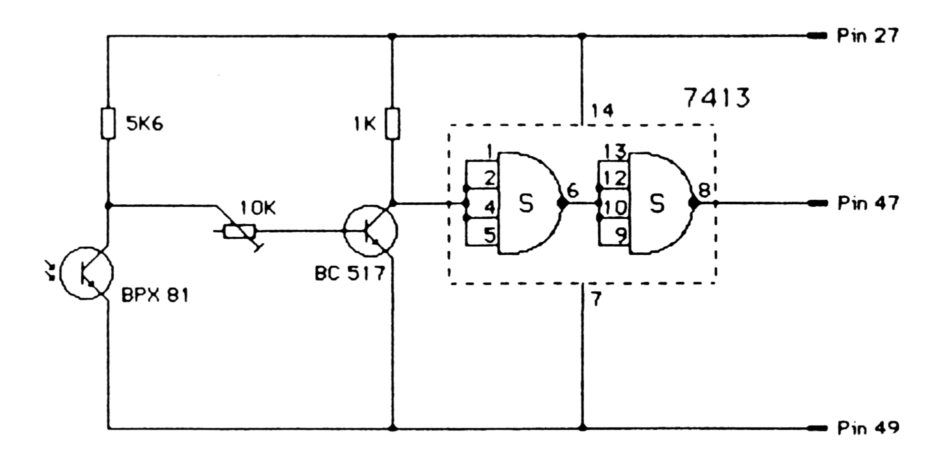


Figure 18 : **Le schéma du circuit du crayon optique**

Liste des composants nécessaires :

TTL-IC 7413
Socle de circuit intégré 14 pôles

Transistor BC 517

Photo-transistor BPX 81

Trimmer 10K

Résistance 1K
Résistance 5K6

Prise pour le port d'extension

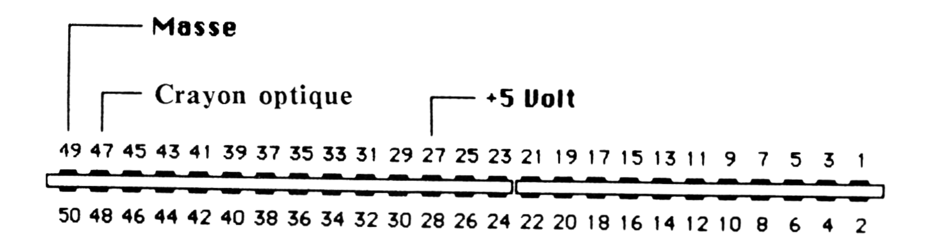


Figure 19 : **Le port d'extension du CPC**

Maintenant que vous êtes parvenu, espérons-le sans encombre, à construire le crayon optique, nous allons vous présenter un petit programme qui vous permettra d'une part de tester un crayon optique et d'autre part de comprendre comment on peut l'interroger dans un programme. Comme nous l'avons déjà indiqué, il s'agit d'un programme de sélection dans un menu. Nous n'avons réalisé que le cadre grossier d'une sélection dans un menu. Nous vous laissons le soin de développer cette partie de programme pour l'intégrer dans un programme utilisateur complet permettant une sélection des options proposées à l'aide du crayon optique.

Le programme marque sur l'écran cinq points de menu parmi lesquels l'utilisateur doit choisir. Il passe immédiatement ensuite à l'interrogation du crayon optique. Si vous marquez l'un des cinq points du menu avec le crayon optique (l'écran couleur doit être réglé sur la luminosité maximale et le trimmer de la carte crayon optique sur la sensibilité maximale) et si vous frappez en même temps n'importe quelle touche au clavier, le programme saute au sous-programme déterminé par la sélection opérée dans le menu. Il exécute alors une boucle d'attente, qui symbolise les opérations qui devraient être effectuées dans ce sous-programme, après quoi il revient à l'interrogation du crayon optique.

```
100 MODE 1
110 INK 0,26
120 INK 1,0
130 BORDER 26
140 '
150 FOR I=1 TO 5
160 LOCATE 15,I*4+1
170 PRINT "OPTION";I
180 NEXT
190 '
200 GOSUB 270
210 '
220 V=INT((P+82)/164)
230 IF INKEY$<>"" THEN CLS:ON V GOSUB 360,390,420,450,480 ELSE GOTO
 200
240 CLS
250 '
260 GOTO 150
270 '
280 'INTERROGATION DU CRAYON OPTIQUE
290 '
300 OUT &BC00,16: PH=INP(&BF00)
310 OUT &BC00,17: PL=INP(&BF00)
320 '
330 P=(256*PH+PL) AND &3FF
340 '
350 RETURN
360 PRINT "SOUS-PROGRAMME 1"
370 FOR I=1 TO 1000:NEXT
380 RETURN
```

```
390 PRINT "SOUS-PROGRAMME 2"
400 FOR I=1 TO 1000:NEXT
410 RETURN
420 PRINT "SOUS-PROGRAMME 3"
430 FOR I=1 TO 1000:NEXT
440 RETURN
450 PRINT "SOUS-PROGRAMME 4"
460 FOR I=1 TO 1000:NEXT
470 RETURN
480 PRINT "SOUS-PROGRAMME 5"
490 FOR I=1 TO 1000:NEXT
500 RETURN
```

La partie caractéristique de la sélection du menu avec le crayon optique est constituée par les lignes de programme 270-330 dans lesquelles est effectuée l'interrogation du crayon optique. Si vous voulez uniquement utiliser ce programme, vous pouvez utiliser telle quelle la méthode d'interrogation du crayon optique présentée dans ce programme. Si par contre vous voulez savoir précisément la signification des registres lus par le programme, nous vous conseillons de consulter le chapitre 9.2.

8.3 COPIE D'ECRAN GRAPHIQUE (graphic hard copy)

C'est justement lorsqu'on se livre à la programmation graphique qu'il arrive fréquemment qu'on souhaite pouvoir garder une trace noir sur blanc, sur l'imprimante, de quelque chose qu'on a réalisé sur l'écran. Il faut donc pouvoir faire une copie d'écran graphique, c'est-à-dire pouvoir lire précisément le contenu de l'écran graphique et en sortir la copie conforme sur l'imprimante.

On distingue deux sortes principales de routines de copie d'écran : la première est la copie d'écran de texte qui consiste à ne sortir sur l'imprimante que les éléments du contenu de l'écran qui sont constitués par des caractères de texte. L'avantage de cette sorte de routines de copie d'écran est qu'elles sont plus rapides, du fait du nombre limité de caractères, que le deuxième type de routines, les routines de copie d'écran graphique. Une routine de copie d'écran graphique se distingue par le fait qu'elle sortira sur l'imprimante le contenu de l'écran, que ce soit du graphisme ou du texte.

Les CPCs présentent en effet l'avantage de permettre de mélanger sur l'écran texte et graphisme.

Malgré son système d'exploitation très puissant, l'AMSTRAD ne dispose pas de routine de copie d'écran. A nous donc de vous doter d'un utilitaire comblant cette lacune. Nous avons naturellement choisi une routine de copie d'écran graphique puisqu'elle vous permettra d'imprimer aussi bien du texte que du graphisme.

Comme la routine de copie d'écran graphique doit tenir le plus grand compte des possibilités de l'imprimante utilisée et comme toutes les imprimantes ne sont pas identiques, il nous a fallu choisir parmi les imprimantes entrant en ligne de compte. Nous avons choisi l'imprimante NLQ 401. Mais comme cette imprimante présente une compatibilité surprenante avec les imprimantes EPSON, le programme tourne également sur les imprimantes EPSON MX/RX/FX ainsi que sur tous les modèles compatibles. Si vous ne possédez pas une imprimante de ce type, cela ne signifie pas forcément que la routine ne tournera pas sur votre imprimante mais seulement que nous ne l'avons pas testée sur votre modèle et que c'est donc à vous de le faire.

Si l'on réfléchit à la tâche que doit remplir une routine de copie d'écran, on voit tout de suite qu'elle ne peut être écrite qu'en langage machine. Pour que ceux d'entre vous qui ne programment pas en langage machine ne soient pas frustrés, nous vous fournissons en plus du code assembleur un programme BASIC de chargement qui écrira la routine de copie d'écran dans la mémoire RAM du CPC. Elle pourra alors y être appelée même à partir du BASIC.

Après que vous ayez lancé le programme de chargement BASIC imprimé ci-dessous, la routine de copie d'écran contenue dans les instructions DATA est placée dans la mémoire RAM du CPC. Elle y figurera alors à partir de l'adresse &A000 et il convient le cas échéant de la protéger contre un empiètement éventuel du BASIC. Pour produire une copie de l'écran actuel, il suffit d'appeler la routine avec CALL &A000.

```
100 'COPIE D'ECRAN GRAPHIQUE
110 '
120 FOR A=&A000 TO &A0BF
130 READ D
```

```
140 POKE A,D
150 Z=Z+D
160 NEXT A
170 '
180 DATA &cd,&ba,&bb,&cd,&e7,&bb,&32,&bd
190 DATA &a0,&cd,&6c,&a0,&21,&8f,&01,&22
200 DATA &be,&a0,&11,&00,&00,&3e,&07,&32
210 DATA &c0,&a0,&cd,&7c,&a0,&0e,&00,&3a
220 DATA &c0,&a0,&47,&e5,&d5,&c5,&cd,&f0
230 DATA &bb,&c1,&d1,&21,&bd,&a0,&be,&e1
240 DATA &37,&20,&01,&a7,&cb,&11,&2b,&2b
250 DATA &10,&e9,&cd,&af,&a0,&79,&cd,&a6
260 DATA &a0,&13,&e5,&21,&7f,&02,&37,&ed
270 DATA &52,&e1,&38,&05,&2a,&be,&a0,&18
280 DATA &cc,&23,&7c,&b5,&c8,&2b,&11,&00
290 DATA &00,&22,&be,&a0,&3e,&07,&bd,&20
300 DATA &b9,&7c,&b4,&20,&b5,&3e,&04,&32
310 DATA &c0,&a0,&18,&ae,&3e,&1b,&cd,&a6
320 DATA &a0,&3e,&31,&cd,&a6,&a0,&00,&00
330 DATA &00,&00,&00,&c9,&e5,&3e,&42,&cd
340 DATA &1e,&bb,&e1,&28,&02,&e1,&c9,&3e
350 DATA &0d,&cd,&a6,&a0,&3e,&0a,&cd,&a6
360 DATA &a0,&3e,&1b,&cd,&a6,&a0,&3e,&4c
370 DATA &cd,&a6,&a0,&3e,&7f,&cd,&a6,&a0
380 DATA &3e,&02,&cd,&a6,&a0,&c9,&cd,&2e
390 DATA &bd,&38,&fb,&cd,&2b,&bd,&c9,&3a
400 DATA &c0,&a0,&fe,&07,&c8,&af,&cb,&11
410 DATA &cb,&11,&cb,&11,&c9,&00,&00,&00
420 '
430 IF Z<>23151 THEN PRINT "ERREUR DANS LES DATAS"
```

Si vous avez des notions de langage machine, le listing assembleur vous permettra de comprendre le mode de fonctionnement de la routine. Vous pourrez vous y référer si vous souhaitez adapter la routine à des imprimantes sur lesquelles elle ne fonctionne pas dans sa version actuelle. Dans votre travail de programmation, n'oubliez surtout jamais que le CPC ne dispose que d'une connexion 7 bits ce qui entraîne quelques complications pour la routine de copie d'écran.

La connexion 7 bits fait par exemple qu'il n'est pas possible de faire imprimer à la fois plus de sept points placés les uns sous

les autres. Or le graphisme du CPC se compose de 200 points verticalement, c'est-à-dire d'un nombre de points qui n'est pas un multiple entier de 7. Les quatre points restants devront donc être traités séparément par la routine de copie d'écran.

Un autre problème dû à la connexion 7 bits concerne la transmission des commandes à l'imprimante. L'activation du mode graphique avec ESC L réclamerait, pour les 640 points d'une ligne de l'écran qui doivent être pris en compte par la routine de copie d'écran, la séquence de commande

```
PRINT #8,CHR$(27);"L";CHR$(128);CHR$(2)
```

Le nombre 128 que contient cette séquence ne peut cependant pas être représenté avec 7 bits et il ne peut de ce fait pas être transmis à l'imprimante. Pour résoudre ce problème, nous avons amputé l'écran artificiellement d'une colonne de points verticale, spécialement pour la routine de copie d'écran. Les derniers points seront tout simplement ignorés lors de l'impression. Le 128 nécessaire pour activer le mode graphique devient donc un 127 qui pourra être transmis à l'imprimante à travers la connexion 7 bits.

```
A000  CD BA BB    CALL  &BBBA        ;Activer mode graphique
A003  CD E7 BB    CALL  &BBE7        ;Déterminer couleur fond
A006  32 BD A0    LD    (&A0BD),A
A009  CD 6C A0    CALL  &A06C        ;Imprimante sur 7/72 pouces
A00C  21 8F 01    LD    HL,&018F     ;Commencer l'impression
A00F  22 BE A0    LD    (&A0BE),HL
A012  11 00 00    LD    DE,&0000     ;En haut à gauche
A015  3E 07       LD    A,&07        ;Avec 7 aiguilles
A017  32 C0 A0    LD    (&A0C0),A
A01A  CD 7C A0    CALL  &A07C        ;Séquence ESC pour graphisme
A01D  0E 00       LD    C,&00        ;C contient masque bits
A01F  3A C0 A0    LD    A,(&A0C0)
A022  47          LD    B,A      ;B=Compteur de séries de points
A023  E5          PUSH  HL
A024  D5          PUSH  DE
A025  C5          PUSH  BC
A026  CD F0 BB    CALL  &BBF0    ;Couleur du point pour (HL,DE)
A029  C1          POP   BC
A02A  D1          POP   DE
A02B  21 BD A0    LD    HL,&A0BD
```

```
A02E   BE          CP    (HL)           ;Couleur point=fond?
A02F   E1          POP   HL
A030   37          SCF                  ;Si point<>paper, alors
A031   20 01       JR    NZ,$ 3 >&A034  ;Fixer CARRY sinon
A033   A7          AND   A              ;Annuler CARRY
A034   CB 11       RL    C              ;Pousser CARRY dans bit le
A036   2B          DEC   HL             ;Plus faible du reg. C,
A037   2B          DEC   HL             ;HL=HL-2, point suivant
A038   10 E9       DJNZ  $-21 >&A023    ;Le tout 7 fois
A03A   CD AF A0    CALL  &A0AF          ;Traitement particulier des
                                         derniers masques bits,
A03D   79          LD    A,C            ;Transférer dans ACCU
A03E   CD A6 A0    CALL  &A0A6          ;Et imprimer
A041   13          INC   DE
A042   E5          PUSH  HL
A043   21 7F 02    LD    HL,&027F       ;Imprime une ligne?
A046   37          SCF
A047   ED 52       SBC   HL,DE
A049   E1          POP   HL
A04A   38 05       JR    C,$ 7 >&A051
A04C   2A BE A0    LD    HL,(&A0BE)
A04F   18 CC       JR    $-50 >&A01D
A051   23          INC   HL             ;Traitement spécial des
A052   7C          LD    A,H            ;4 Derniers
A053   B5          OR    L
A054   C8          RET   Z
A055   2B          DEC   HL
A056   11 00 00    LD    DE,&0000       ;Préparation de la ligne
A059   22 BE A0    LD    (&A0BE),HL     ;D'impression suivante
A05C   3E 07       LD    A,&07
A05E   BD          CP    L              ;Dernière série de 7?
A05F   20 B9       JR    NZ,$-69 >&A01A
A061   7C          LD    A,H
A062   B4          OR    H
A063   20 B5       JR    NZ,$-73 >&A01A
A065   3E 04       LD    A,&04          ;Alors plus que 4 lignes
A067   32 C0 A0    LD    (&A0C0),A
A06A   18 AE       JR    $-80 >&A01A
A06C   3E 1B       LD    A,&1B          ;Pour NLQ/MX/RX/FX
A06E   CD A6 A0    CALL  &A0A6          ;ESC A 7, pour obtenir le
A071   3E 31       LD    A,&31          ;Bon saut de ligne
A073   CD A6 A0    CALL  &A0A6
```

```
A076   00          NOP
A077   00          NOP
A078   00          NOP
A079   00          NOP
A07A   00          NOP
A07B   C9          RET
A07C   E5          PUSH  HL             ;Touche DEL enfoncée?
A07D   3E 42       LD    A,&42          ;Si oui, alors arrêt
A07F   CD 1E BB    CALL  &BB1E
A082   E1          POP   HL
A083   28 02       JR    Z,$ 4 >&A087   ;DEL pas enfoncée
A085   E1          POP   HL             ;Manipuler la pile
A086   C9          RET                  ;Pour parvenir au RET
A087   3E 0D       LD    A,&0D          ;Sortir CR/LF
A089   CD A6 A0    CALL  &A0A6
A08C   3E 0A       LD    A,&0A
A08E   CD A6 A0    CALL  &A0A6
A091   3E 1B       LD    A,&1B          ;ESC L 127 2=Graphisme
A093   CD A6 A0    CALL  &A0A6          ;Avec 639 points
A096   3E 4C       LD    A,&4C
A098   CD A6 A0    CALL  &A0A6
A09B   3E 7F       LD    A,&7F
A09D   CD A6 A0    CALL  &A0A6
A0A0   3E 02       LD    A,&02
A0A2   CD A6 A0    CALL  &A0A6
A0A5   C9          RET
A0A6   CD 2E BD    CALL  &BD2E          ;Imprimante BUSY?
A0A9   38 FB       JR    C,$-3 >&A0A6
A0AB   CD 2B BD    CALL  &BD2B          ;Imprimer caractère
A0AE   C9          RET
A0AF   3A C0 A0    LD    A,(&A0C0)      ;Traitement des quatre
A0B2   FE 07       CP    &07            ;Dernières lign. de
                                         points
A0B4   C8          RET   Z
A0B5   AF          XOR   A
A0B6   CB 11       RL    C              ;Pousser trois fois 0
A0B8   CB 11       RL    C              ;Dans le registre C
A0BA   CB 11       RL    C              ;A travers le CARRY
A0BC   C9          RET
A0BD   00          DEFB  00
A0BE   00 00       DEFW  0000
A0C0   00          DEFB  00
```

8.4 BANDE-TEXTE

Si vous souhaitez donner à vos paroles un poids particulier dans certaines occasions, le programme que nous vous proposons vous permettra de le faire très efficacement. Il produit sur l'imprimante une bande-texte avec des lettres énormes. Vous pouvez déterminer librement le texte que vous voulez voir sortir. Chaque lettre peut avoir une hauteur allant jusqu'à 80 caractères d'impression normaux. Les lettres sont sorties sur l'imprimante couchées par rapport au sens normal de l'écriture de l'imprimante. Comme le programme peut dévorer des quantités respectables de papier suivant la longueur du texte que vous avez entré, il est absolument recommandé d'utiliser du papier continu.

Le mode d'emploi du programme est aussi simple que possible : après qu'il ait été lancé avec RUN, on demande à l'utilisateur d'entrer le texte. Les lettres tapées sont écrites dans la première ligne de l'écran. Si vous avez fait une erreur ou si vous voulez modifier le texte déjà entré, vous pouvez effacer le texte caractère par caractère avec la touche DEL.

Une fois que le texte a été entré, vous pouvez déclencher la sortie sur imprimante en actionnant la touche RETURN. Si le texte occupe entièrement les 80 caractères de la première ligne, la sortie sur imprimante est automatiquement déclenchée dès que vous dépassez le dernier emplacement de caractère.

Pendant la sortie sur imprimante, le programme examine les points d'image fixés dans la première ligne de l'écran et il les envoie, considérablement agrandis, sur l'imprimante. Chaque point d'image marqué devient une case de 10 fois 5 dièzes. Cet examen fait que le contenu de la première ligne de l'écran peut être imprimé quel qu'il soit. Ce programme ne vous limite donc pas à l'emploi des caractères du jeu de caractères intégré du CPC et vous pouvez parfaitement utiliser votre propre jeu de caractères. Vous pouvez ainsi obtenir des résultats très plaisants si vous utilisez ce programme avec un jeu de caractères modifié et de petits éléments graphiques, de la taille d'un caractère, qui puissent être intégrés dans le texte.

```
100 '*********************************************************
110 '***                                                   ***
120 '***          BANDE-TEXTE  JST 5.8.1986                ***
130 '***                                                   ***
140 '*********************************************************
150 '
160 MODE 2
170 '
180 CLS
190 '
200 DIM MATRICE(639,7)
210 '
220 START=1
230 POINT$=STRING$(10,"#")
240 ESPACE$=SPACE$(10)
250 '
260 'ENTREE DU TEXTE
270 '
280 LOCATE 1,10
290 PRINT "ENTREZ LE TEXTE:"
300 '
310 IF START<=1 THEN START=1
320 '
330 FOR POSITION=START TO 80
340 '
350 LOCATE POSITION,1
360 PRINT " ";
370 '
380 CARACTERE$=INKEY$
390 IF CARACTERE$="" THEN 380
400 IF ASC(CARACTERE$)=127 THEN START=POSITION-1:GOTO 310
410 IF ASC(CARACTERE$)=13 THEN GOTO 480
420 '
430 LOCATE POSITION,1
440 PRINT CARACTERE$;
450 '
460 NEXT POSITION
470 '
480 'PRODUIRE BANDE-TEXTE
490 '
500 FIN=((POSITION-1)*8)-8
510 IF FIN<0 THEN END
```

```
520 '
530 FOR INDEX=0 TO FIN STEP 8
540 '
550 'LIRE UN CARACTERE SUR L'ECRAN
560 '
570 FOR COLONNE=INDEX TO INDEX+7
580 FOR LIGNE=0 TO 7
590 '
600 MATRICE(COLONNE,LIGNE)=TEST(COLONNE,399-LIGNE*2)
610 '
620 NEXT LIGNE
630 NEXT COLONNE
640 '
650 'SORTIE SUR IMPRIMANTE
660 '
670 FOR LIGNE=INDEX TO INDEX+7
680 FOR MULTI=1 TO 5
690 FOR COLONNE=7 TO 0 STEP -1
700 '
710 IF MATRICE(LIGNE,COLONNE)=1 THEN PRINT #8,POINT$; ELSE PRINT #8
,ESPACE$;
720 '
730 NEXT COLONNE
740 '
750 PRINT #8
760 '
770 NEXT MULTI
780 NEXT LIGNE
790 '
800 NEXT INDEX
```

Description du programme :

100-240 Définitions, dimensionnements et pré-affectations. MODE 2 est sélectionné pour obtenir une sortie sur écran de 80 colonnes. La variable doublement indicée MATRICE est dimensionnée à 640 fois 8 cases. Elle recevra dans le déroulement du programme les matrices de caractère lues sur l'écran.

La variable auxiliaire START est fixée sur la valeur de départ 1. Elle indique la position du début de la sortie de la bande-texte sur l'écran.

Les variables POINT$ et ESPACE$ se voient affecter respectivement dix dièzes et dix espaces. Ces variables seront utilisées pour la sortie, en format agrandi, des matrices de caractère sur l'imprimante.

250-460 *Entrée du texte*

Cette partie du programme sert à sortir sur l'écran et éventuellement à modifier le texte qui devra apparaître plus tard sur la bande-texte. Le programme demande en ligne 290 d'entrer le texte. L'entrée du texte se fait lettre par lettre dans une boucle FOR-TO-NEXT qui va de la ligne 330 à la ligne 460. La lettre sur la position actuelle sur la bande-texte (POSITION) est d'abord effacée (ce qui est important pour l'édition) et une entrée au clavier est ensuite prise en compte. Cette entrée répond à l'un des quatre cas suivants :

1. Aucune touche n'a été actionnée, aller chercher une nouvelle valeur au clavier (INKEY$).

2. Le caractère entré a le code 127 (touche DEL), ce qui fait que la position d'écriture sur la bande-texte est décrémentée et affectée à la variable START. On saute ensuite à la ligne 310 où l'on détermine si le bord gauche a été atteint en décrémentant POSITION. Si c'est le cas, le bord gauche est pris comme nouvelle position (START=1). La boucle FOR-TO-NEXT d'entrée des caractères est alors à nouveau parcourue avec la position de départ ainsi manipulée. C'est maintenant qu'on comprend pourquoi le caractère dans la position d'écriture actuelle est systématiquement effacé.

3. La touche RETURN a été actionnée (Code 13). Le programme considérera donc que l'entrée de la bande-texte est terminée et quittera la partie ENTREE DU TEXTE du programme (GOTO 480).

4. L'entrée d'un autre caractère entraîne la sortie de ce caractère dans la position d'écriture actuelle sur la bande-texte. La boucle FOR-TO-NEXT est normalement parcourue.

470-Fin *Produire la bande-texte*
Cette partie du programme lit, point par point, le texte de la bande-texte dans la mémoire écran du CPC. Ce texte est stocké, toujours point par point, dans la variable indicée MATRICE(X,Y). C'est d'après cette variable que sera réalisée la sortie sur l'imprimante connectée. Ce n'est qu'avec cette méthode, qui peut sembler compliquée de prime abord, que l'on peut envoyer n'importe quel caractère sur l'imprimante, même si ce caractère a été redéfini.

La ligne 500 déduit la position X du curseur graphique de la dernière position dans laquelle a été écrit un caractère de la bante-texte (colonne gauche de la matrice de caractère). Si la valeur ainsi calculée est inférieure à zéro, cela signifie qu'aucun texte n'a été entré et le programme rencontre END. Une boucle FOR-TO-NEXT, qui va de la ligne 530 à la fin du programme, englobe tout le reste. Dans cette boucle sont produites sous le nom de "INDEX" des valeurs qui représentent chaque fois la plus petite position X d'un caractère de la bande-texte par rapport à l'écran graphique. Dans cette boucle FOR-TO-NEXT figurent deux sections du programme, la première étant appelée LIRE UN CARACTERE SUR L'ECRAN et la seconde SORTIE SUR IMPRIMANTE.

LIRE UN CARACTERE SUR L'ECRAN se compose de deux boucles FOR-TO-NEXT imbriquées qui permettent de lire la matrice d'un caractère situé sur la position définie par INDEX. Les deux boucles imbriquées définissent des positions de l'écran dans lesquelles le numéro du crayon de couleur utilisé peut être déterminé avec la fonction BASIC TEST. Ce numéro est affecté à la variable indicée MATRICE (COLONNE,LIGNE).

La partie SORTIE SUR IMPRIMANTE du programme comporte trois boucles FOR-TO-NEXT imbriquées. Deux de ces boucles produisent, comme dans LIRE UN CARACTERE SUR L'ECRAN, une clé d'accès à la variable doublement indicée MATRICE. L'accès est ici modifié en fonction des exigences de la sortie sur imprimante. La véritable sortie sur l'imprimante se fait en ligne 710. POINT$ est sorti si le point d'image correspondant de la sortie sur écran était mis. Si ce point d'image n'était pas mis, c'est ESPACE$ qui est sorti. Une fois qu'une ligne entière a été sortie sur l'imprimante, la troisième boucle FOR-TO-NEXT se charge de faire sortir la même ligne cinq fois au total. On obtient ainsi un tracé équilibré des caractères.

Une fois que la boucle FOR-TO-NEXT qui englobe tout a atteint sa valeur finale, c'est que la bande-texte a été entièrement sortie sur l'imprimante.

9. La programmation en langage machine sur le CPC

Comme vous avez pu vous en rendre compte jusqu'ici, le BASIC AMSTRAD offre une quantité de fonctions graphiques qui permettent de résoudre un nombre étonnant de problèmes de programmation. La vitesse d'exécution des programmes BASIC présentés jusqu'ici était presque toujours satisfaisante bien que nous devions travailler avec un langage interprété et 16 K octets de RAM vidéo. Malgré tout, il y aura toujours de nombreux problèmes graphiques qui dépasseront largement les capacités de l'interpréteur BASIC. Le graphisme tridimensionnel constituait déjà un bon exemple d'un domaine où le système BASIC est vraiment à la limite de la saturation. Mais il y a des programmes qui ne sont même pas réalisables à partir du niveau du BASIC. Songez par exemple au développement d'un générateur de sprites (objets animés), à un scrolling (glissement de l'écran) progressif ou à la gestion éclair de deux pages écran indépendantes.

La programmation en langage machine peut au contraire vous permettre d'exploiter pleinement les possibilités du matériel électronique dont nous disposons et de réduire considérablement les temps d'exécution de nos programmes ou de nos modules de programme grâce à une programmation spécifiquement axée sur le graphisme. Des programmes tels que Superpaint montrent bien qu'on peut faire sur le CPC des choses très intéressantes qui étaient au départ réservées à des ordinateurs dix fois plus chers.

9.1 LE COEUR DU CPC - L'UNITE CENTRALE Z80

Cette partie est conçue en premier lieu pour les lecteurs qui ont des notions générales de programmation en langage machine ou qui veulent rafraîchir leurs connaissances sur le Z80. Il ne nous est malheureusement pas possible ici de vous fournir une introduction détaillée au langage machine ni de reproduire les tables d'instructions complètes du langage machine. Ce n'est pas le but du présent ouvrage et cela ne pourrait qu'augmenter encore ses dimensions déjà imposantes. Nous allons simplement vous présenter l'organisation de base du processeur pour que vous puissiez mieux mettre à profit les listings assembleur que nous vous proposerons par la suite.

Nous ne pouvons que renvoyer ceux d'entre vous qui souhaiteraient s'initier au langage machine à l'abondante littérature technique traitant de ce sujet. Citons notamment **"Le langage machine sur l'Amstrad CPC" de DATA BECKER - MICRO APPLICATION.**

L'unité centrale Z80 a été conçue en 1974 par la société américaine ZILOG. Il s'agit d'un développement logique du microprocesseur 8080, qui avait fait ses preuves jusque là et qui représente l'élément de base du système d'exploitation CP/M (qui est également fourni avec votre CPC). Il y a une compatibilité ascendante entre le 8080 et le Z80, c'est-à-dire que tous les programmes écrits sur un 8080 peuvent être exécutés sans problème sur un Z80 mais pas l'inverse. C'est pourquoi pratiquement tous les constructeurs préfèrent actuellement employer les processeurs Z80, beaucoup plus puissants, dans leurs systèmes CP/M.

AMSTRAD a opté pour l'emploi d'un Z80-A, un microprocesseur Z80 doté d'une fréquence d'horloge de 4 MHz. Il appartient, comme le 8080, à la catégorie des processeurs 8 bits.

Le jeu de registres du Z80

Le Z80 possède au total 22 registres. Il s'agit du compteur de programme, du pointeur de pile, de registres d'index ainsi que de deux accumulateurs et deux registres flags. Cela peut sembler curieux de prime abord mais la raison en est très simple. Le Z80 dispose de deux jeux de registres complets qu'il est possible de commuter avec une instruction spéciale. Ce double jeu de registres rend l'unité centrale Z80 extrêmement puissante car le transfert de données à l'intérieur du processeur est toujours plus rapide qu'un accès à la RAM.

Voici maintenant une présentation de tous les registres du microprocesseur Z80.

Les registres 8 bits

Les registres A, B, C, D, E, F, H et L représentent le jeu de registres de base de notre microprocesseur, jeu de registres qui existe en double exemplaire.

Ces registres permettent de stocker chacune des valeurs comprises entre 0 et 255. A cet égard, les registres A et F ont des fonctions particulières par rapport aux autres registres.

A (Accumulateur) :

L'accumulateur ou accu reçoit toujours le résultat d'une opération arithmétique ou logique sur 8 bits. Il peut naturellement aussi être utilisé comme mémoire de stockage provisoire.

F (Registre flag) :

Les différents bits du registre flag sont modifiés par le processeur après chaque opération arithmétique ou logique. Ils indiqueront par exemple ainsi si un débordement s'est produit dans un registre ou si le résultat d'une opération est nul. On pourra ainsi prendre des décisions logiques à l'aide du registre F.

Registres 8 bits spéciaux

Les registres 8 bits restants, I et R, ne sont pas utilisés par le système d'exploitation des ordinateurs CPC. Bien que ces registres soient normalement destinés à des fonctions spéciales, ils peuvent parfaitement être utilement employés dans nos programmes.

R (Reg. Refresh) :

Le registre R fait exécuter un refresh (régénération de la mémoire) automatique par l'unité centrale Z80. A cet effet, les bits relatifs 0 à 6 sont incrémentés en permanence. Le bit relatif 7 n'est pas affecté. Il peut être manipulé par l'utilisateur en fonction de ses besoins. Le registre refresh peut être par ailleurs "détourné" de sa fonction pour être employé dans une application. Si vous voulez par exemple réaliser un jeu vidéo, vous aurez très probablement besoin de nombres aléatoires. Comme l'état du registre refresh ne peut jamais être prévu à l'avance, il peut constituer un excellent générateur de hasard.

I (Rg. d'interruption) :

Le registre d'interruption 8 bits. N'est utilisé qu'en liaison avec le mode d'interruption IM2. Il représente alors l'octet fort d'une page mémoire de 256 octets de long dans laquelle vous pouvez stocker jusqu'à 128 vecteurs d'interruption. Le registre I peut être employé simplement comme une mémoire rapide de 8 bits si IM2 n'est pas employé dans votre programme. L'accès au registre d'interruption est en effet plus lent que l'accès à un registre normal mais il reste toujours plus rapide qu'un accès direct à la mémoire.

Doubles registres 16 bits

Le Z80 est un des rares processeurs 8 bits à disposer de registres 16 bits. Ces registres 16 bits présentent des avantages de vitesse énormes. Ils nous permettent par exemple d'additionner avec une seule instruction deux nombres compris entre 0 et 65535. Même une multiplication par 2 sur 16 bits ne posera aucun problème!

Le Z80 dispose de deux différentes sortes de registres 16 bits :

- les vrais registres 16 bits et
- les registres 16 bits simulés, qu'on appelle les doubles registres

Les doubles registres sont les couples de registres 8 bits BC, DE et HL. Comme ils ne sont au fond que la réunion de registres élémentaires 8 bits, ces registres sont heureusement également présents en double exemplaire (double jeu de registres!). Le registre HL joue pour ce mode d'adressage le rôle d'accumulateur 16 bits. Le registre flag garde la même signification.

Les registres d'index 16 bits

L'unité centrale nous offre deux autres registres 16 bits (IX et IY) pour l'adressage indexé. Ces registres peuvent être dotés d'un offset (adresse de distance) supplémentaire de 8 bits.

Cet offset sera automatiquement additionné au contenu du registre d'index correspondant. Cet offset est écrit en complément à deux de sorte qu'il permet également d'additionner des adresses de distance négatives.

Les registres 16 bits spéciaux

Les registres 16 bits restants sont modifiés entre autre par le Z80 lui-même.

SP (pointeur de pile) :

> Le registre SP (Stack Pointer) contient l'adresse d'une zone de la mémoire protégée dans laquelle peuvent être stockées provisoirement des données 16 bits. L'unité centrale Z80 place dans cette zone les adresses de retour de toutes les instructions CALL. Le transfert d'un mot (16 bits) de données dans cette zone entraîne un double décrément du pointeur de pile.

PC (Compteur de programme) :

> Le registre PC (Program Counter) contient toujours l'adresse de l'instruction devant être actuellement traitée par l'unité centrale. Après exécution de l'instruction, le processeur ajoute automatiquement la longueur en octets du code d'opération actuellement traité et de l'opérande possible au contenu du registre PC. Il peut également être modifié directement par l'utilisateur à travers une instruction jump ou call.

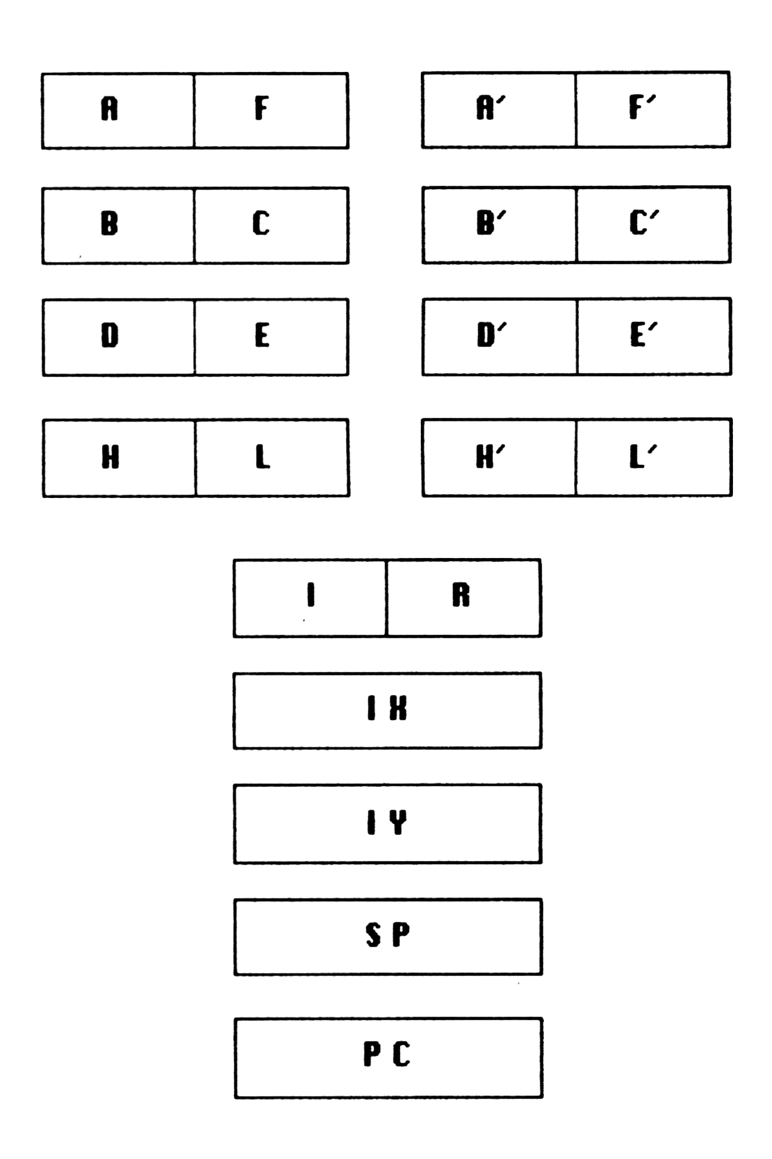


Figure 20 : Le jeu de registres du Z80

9.2 LE CONTROLEUR VIDEO

Le contrôleur vidéo gère toutes les formes de sortie sur l'écran. Sur tous les CPCs, il s'agit du HD 6845, qui est un représentant remarquable de sa catégorie et qu'on retrouve dans des systèmes beaucoup plus chers.

Le contrôleur peut être adressé par ses ports I/O (entrée/sortie) selon la méthode traditionnelle du Z80. Cet adressage de port présente l'avantage, par rapport à la méthode Memory mapped, de ne pas mobiliser une zone de RAM supplémentaire. Puisque nous parlons de l'adressage de port : vous savez que le Z80 met 256 ports I/O à la disposition de l'utilisateur. Or le système d'exploitation de l'AMSTRAD CPC gère 65536 canaux d'entrée/sortie. Comment est-ce possible? C'est que les constructeurs du Z80 ont utilisé une astuce très simple mais très efficace pour permettre un emploi universel de leur unité centrale. Celle-ci comporte en effet des instructions qui permettent de sélectionner le port I/O voulu au moyen d'un mot de données (=16 bits). Il faut pour cela charger l'adresse de port voulue dans le registre BC. Ce mode d'adressage a cependant également des inconvénients. C'est qu'en effet il n'est pas possible d'utiliser dans ce mode toutes les instructions de port disponibles.

On dispose pour l'adressage du HD 6845 de 18 registres de données et d'un registre d'adresse. Avant d'écrire ou lire un registre de données, il faut placer son numéro dans les 5 bits inférieurs du registre d'adresse. Les registres sont adressés ici à travers les ports suivants :

Registre d'adresse : &BC00
Registre de données : &BD00

Une programmation appropriée du contrôleur vidéo permet d'obtenir une programmation graphique nettement plus rapide, plus élégante et plus simple. C'est ainsi par exemple que la connexion du crayon optique ou le scrolling du contenu de l'écran deviennent très simples en programmant directement le contrôleur vidéo.

Le jeu de registres du contrôleur vidéo

AR Le registre d'adresse (WRITE 5 bits)
Il faut charger le numéro du registre de données à adresser dans le registre d'adresse. Il accepte les numéros de registre 0 à 17. Les valeurs 18 à 31 sont ignorées. On ne peut qu'écrire dans le registre d'adresse.

R0 Horizontal Total (WRITE 8 bits)
Ce registre contient le nombre de caractères par ligne complète. Le terme de "complète" ne fait pas référence aux caractères visibles mais à la valeur attendue par le système d'exploitation. Comme il tient compte également des caractères pour le bord et le retour du faisceau, la valeur doit être de 1,5 fois le nombre de caractères vraiment représentés. On ne peut qu'écrire dans le registre R0.

R1 Horizontal Displayed (WRITE 8 bits)
R1 contient le nombre réel (visible) de caractères par ligne. Cette valeur doit toujours être inférieure à la valeur en R0. On ne peut qu'écrire dans le registre R1.

R2 Horizontal Sync-Position (WRITE 8 bits)
Ce registre permet d'obtenir un effet très particulier. Il indique en effet le moment de l'impulsion HSync. Une diminution de sa valeur décale l'ensemble de l'image du moniteur vers la droite, une augmentation la décalera au contraire vers la gauche. On ne peut qu'écrire dans le registre R2.

R3 Sync Width (WRITE 4 bits)
Les bits relatifs 0 à 3 déterminent la largeur des impulsions HSync et VSync. Les bits restants ne sont pas utilisés. On ne peut qu'écrire dans le registre R3.

R4 Vertical Total (WRITE 7 bits)
Le contenu de R4 fixe le nombre de lignes de grille par image. Il détermine également si la fréquence de régénération de l'image doit être synchronisée en intervalles de 50 ou de 60 Hertz. On ne peut qu'écrire dans le registre R4.

R5 Vertical Total Adjust (WRITE 6 bits)
Les 6 bits inférieurs de ce registre sont chargés de l'ajustage (tuning) de la fréquence de régénération de l'image. On ne peut qu'écrire dans le registre R5.

R6 Vertical Displayed (WRITE 7 bits)
Les bits relatifs 0 à 7 représentent le nombre réel de lignes de grille sur l'affichage. La valeur entrée ici doit toujours être inférieure à celle en R4. On ne peut qu'écrire dans le registre R6.

R7 Vertical Sync-Position (WRITE 7 bits)
Le contenu de R7 permet, de façon semblable à R2, d'obtenir un décalage vertical de l'ensemble de l'image. Il détermine le moment de l'impulsion VSync. Si cette valeur est diminuée, l'image du moniteur sera décalée vers le bas, si elle est augmentée l'image sera décalée vers le haut. On ne peut qu'écrire dans le registre R7.

R8 Interlace (WRITE 2 bits)
Les bits relatifs 0 et 1 déterminent si la représentation de l'écran doit se faire avec ou sans interlace (méthode de saut de ligne). On ne peut qu'écrire dans le registre R8.

R9 Maximum Register Adress (WRITE 5 bits)
R9 contient le nombre de lignes de grille utilisées pour représenter un caractère. On ne peut qu'écrire dans le registre R9.

R10 Cursor Start Raster (WRITE)
R10 est chargé de la gestion du curseur. Les cinq bits inférieurs fixent la première ligne de la grille pour le dessin du curseur. Les bits 5 et 6 fixent en outre l'un des quatre modes curseur, d'après la définition suivante :

Bits 6	**5**	
0	0	curseur normal
0	1	pas de curseur
1	0	le curseur clignote environ 3 fois par seconde
0	1	le curseur clignote environ 1,5 fois par seconde

On ne peut qu'écrire dans le registre R10.

R11 Cursor End Raster (WRITE 5 bits)
R11 est chargé de la gestion du curseur. Les cinq bits inférieurs du registre indiquent la dernière ligne de la grille pour le dessin du curseur. On ne peut qu'écrire dans le registre R11.

R12 Start Adress High (READ/WRITE 6 bits)
Le registre R12 nous ouvre une possibilité très intéressante. Les 6 bits inférieurs fixent l'octet fort de l'adresse du début de la zone vidéo dans la zone d'adresses de 16 K du contrôleur vidéo. On peut aussi bien lire qu'écrire dans le registre R12. Lorsqu'on lit ce registre, les deux bits les plus élevés sont toujours dans un état low.

R13 Start Adress Low (READ/WRITE 8 bits)
Le registre R13 fixe l'octet faible de l'adresse du début de la zone vidéo dans la zone d'adresses de 16 K du contrôleur vidéo. On peut aussi bien lire qu'écrire dans le registre R13.

R14 Cursor High (READ/WRITE 6 bits)
Les bits relatifs 0 à 5 représentent l'octet fort de la position actuelle du curseur. On peut aussi bien lire qu'écrire dans le registre R14.

R15 Cursor Low (READ/WRITE 8 bits)
Ce registre contient l'octet faible de la position actuelle du curseur. On peut aussi bien lire qu'écrire dans le registre R15.

R16 Light-Pen-High (READ 6 bits)
R16 est un registre de crayon optique qui peut être utilisé en liaison avec R17. Il contient, après chaque impulsion positive strobe, l'octet fort de l'adresse de l'écran actuellement activée. On ne peut que lire le registre R16.

R17 Light-Pen-Low (READ 8 bits)
Ce registre est un registre de crayon optique qui peut être utilisé en liaison avec R16. Il contient, après chaque impulsion positive strobe, l'octet faible de l'adresse de l'écran actuellement active. On ne peut que lire le registre R17.

9.3 LE GATE ARRAY

Le Gate Array est un composant important de votre ordinateur CPC. Il remplit en effet non seulement des fonctions importantes de coordination système mais aussi de nombreuses fonctions ayant trait au graphisme. Il s'agit par exemple de la production des couleurs ou du changement de mode écran.

Comme le contrôleur vidéo, le jeu de registres du Gate Array est adressé à travers les ports d'entrée/sortie du Z80. L'adresse qu'il faut utiliser ici est &7F00. Elle vous permet de commander trois registres différents. On ne peut qu'écrire dans ces registres. Si vous avez toutefois besoin de messages de réponse dans vos programmes, vous devez placer chaque manipulation du Gate Array dans des variables. Cette méthode ne fonctionne naturellement que pour vos routines personnelles.

Les registres du Gate Array

PEN :

Le registre PEN est chargé de l'affectation des couleurs. On doit y charger l'offset de l'adresse qui doit recevoir un nouveau code couleur. Cet offset correspond au paramètre PEN du BASIC AMSTRAD.

Les deux bits supérieurs de l'octet d'adresse sélectionnent le registre PEN. Ils doivent être tous deux en état low (0). Les bits relatifs 0 à 3 contiennent l'offset de l'adresse à modifier (0 à 15). Le cinquième bit n'est pas utilisé alors que le bit numéro 4 a une signification particulière. S'il est mis, les 4 bits inférieurs seront ignorés et c'est la valeur de l'instruction OUT venant immédiatement à la suite et adressant le registre INK qui sera interprétée comme nouvelle couleur de cadre.

INK :

Ce registre ne peut être utilisé qu'en liaison avec le registre PEN. Il charge un nouveau code couleur dans l'adresse qui a été sélectionnée avec le registre PEN. Le code couleur est indentique aux codes couleur de l'instruction BASIC INK.

Les bits 7 et 6 de l'octet d'adresse sélectionnent le registre INK. Le bit 7 doit toujours être en état low et le bit 6 en état high. Les 5 bits inférieurs contiennent le code de la couleur à charger (0 à 26). Le bit relatif numéro 5 n'est pas utilisé.

MFR :

Le registre multifonctions a, comme son nom l'indique, différentes tâches. On peut y écrire lorsque le bit relatif 7 de l'octet d'adresse est en état high (1) et le bit 6 en état low (0). Le bit numéro n'est pas utilisé. Les autres bits permettent de sélectionner la fonction voulue d'après la définition suivante :

Bit 4 : effacer le compteur de V-Sync
Bit 3 : activer/désactiver la ROM (&C000 à &FFFF)
Bit 2 : activer/désactiver la ROM (&0000 à &3FFF)
Bit 1 : changer le mode écran
Bit 0 : changer le mode écran

Si le bit 4 est mis lors de la sortie de l'octet d'adresse, le pointeur système, qui indique le vecteur d'interruption devant être traité, est fixé sur zéro. Le vecteur d'interruption de plus grande priorité sera donc traité lors de la prochaine interruption.

Les bits relatifs 2 et 3 déterminent par leur état actuel la configuration ROM/RAM. Si un des deux bits est mis, c'est la banque de RAM correspondante qui sera lue, sinon ce sera la ROM.

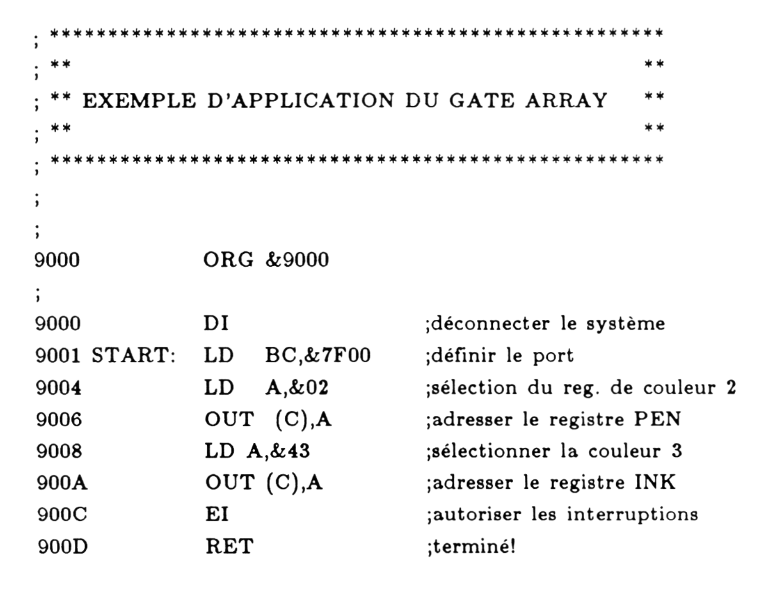
Les deux derniers bits définissent les modes écran. Les quatre différentes combinaisons possibles permettent d'appeler tous les modes écran ainsi qu'un mode spécial, d'après la définition suivante :

Bit 1	**Bit 0**	**Fonction**
0	0	Mode 0 (160*200 points/16 couleurs/clignot.)
0	1	Mode 1 (320*200 points/4 couleurs/clignot.)
1	0	Mode 2 (640*200 points/2 couleurs/clignot.)
1	1	Mode 0 (160*200 points/16 couleurs)

Voici maintenant une récapitulation de la définition des bits supérieurs de tous les registres :

Bit 7	Bit 6	Fonction
0	0	Charger l'offset d'adresse de couleur dans le registre PEN
0	1	Charger le code couleur dans le registre INK
1	0	Initialiser la fonction du registre MFN
0	0	Utilisé par le système pour la commutation de mémoire sur le 6128

Voilà, c'est tout! Mais comment peut-on employer utilement le Gate Array? Eh bien si vous voulez accéder en premier lieu, dans vos programmes machine, au routines de la ROM, il sera en général recommandé de pouvoir accéder à toutes les routines disponibles. Nous vous expliquerons plus loin comment accéder, pour ce faire, à la ROM à travers des vecteurs. Mais si vous voulez accéder à vos propres routines, pour des raisons de gain de place mémoire ou de vitesse, il est indispensable de bien connaître les règles que nous venons de vous exposer. L'exemple suivant vous montre la simplicité d'emploi du Gate Array :



```
; **************************************************
; **                                              **
; ** EXEMPLE D'APPLICATION DU GATE ARRAY          **
; **                                              **
; **************************************************
;
;
9000            ORG &9000
;
9000            DI              ;déconnecter le système
9001 START:     LD   BC,&7F00   ;définir le port
9004            LD   A,&02      ;sélection du reg. de couleur 2
9006            OUT  (C),A      ;adresser le registre PEN
9008            LD A,&43        ;sélectionner la couleur 3
900A            OUT (C),A       ;adresser le registre INK
900C            EI              ;autoriser les interruptions
900D            RET             ;terminé!
```

Cette petite routine charge le code de couleur 3 dans le registre d'adresse de couleur numéro 2. Elle correspond strictement à l'instruction BASIC INK 2,3. Si vous voulez essayer de réaliser ce même exemple avec les instructions BASIC

```
OUT &7F00,2:OUT &7F00,&43        <RETURN>
```

vous aurez une surprise. Tout ce que vous obtiendrez, ce sera un éclair rapide de la couleur sélectionnée. Ce phénomène est dû au fait que le système d'exploitation représente systématiquement toutes les couleurs du CPC au moyen de la gestion des interruptions. Lors de chaque interruption, le système d'exploitation transfère automatiquement un des deux codes couleur dans le Gate Array. Comme les deux couleurs utilisées pour le clignotement sont normalement identiques, l'utilisateur ne remarque pas en principe ce changement incessant de couleur. Mais si on entreprend, comme dans l'exemple qui nous intéresse, une manipulation de couleur au niveau du BASIC, à l'aide du Gate Array, on peut voir la nouvelle couleur jusqu'à la prochaine interruption. C'est ce qui produit ce bref éclair sur votre écran.

9.4 PROGRAMMATION GRAPHIQUE DU Z80

L'unité centrale Z80 est un processeur orienté registre. Cela signifie que la vitesse d'exécution de nos programmes sera d'autant plus élevée que nous utiliserons plus de registres. Comme beaucoup de routines graphiques devront être exécutées aussi vite que possible, il s'agit d'utiliser au mieux les avantages offerts par le Z80. C'est notamment dans le domaine de la programmation graphique qu'il importe d'éviter de gaspiller le moindre cycle d'horloge car c'est le seul moyen d'obtenir sur l'écran des mouvements sans à-coups.

Nous avons vu au chapitre 9.1 que le processeur Z80 comporte deux jeux de registres principaux. Le second jeu de registres est toutefois utilisé par le système d'exploitation du CPC et nous ne pouvons malheureusement pas l'utiliser dans nos routines propres à cause de cela. Si vous tentez de le faire malgré tout, le "plantage" du système sera inévitable.

Ce simple fait limite fortement la programmation graphique essentiellement axée sur la vitesse. Une petite astuce peut cependant vous permettre de tromper le système d'exploitation AMSTRAD et d'accéder malgré tout au second jeu de registres convoité.

La routine suivante permet d'éviter le plantage du système :

```
; **********************
; **                  **
; ** USE 2ND REG      **
; **                  **
; **********************
;
;
DI              ;DECONNECTER LE SYSTEME
EX    AF,AF'    ;ALLER CHERCHER LES SECONDS REGISTRES
EXX             ;     "
PUSH  AF        ;ET LES SAUVER
PUSH  BC        ;     "
PUSH  DE        ;     "
PUSH  HL        ;     "
;
CALL PROGRAM    ;APPEL DU PROGRAMME PRINCIPAL
;
POP   HL        ;ALLER CHERCHER LES REGISTRES SAUVES
POP   DE        ;     "
POP   BC        ;     "
POP   AF        ;     "
EXX             ;ALLER CHERCHER LES REGISTRES DE TRAVAIL
EX    AF,AF'    ;     "
EI              ;"REACTIVER" LE SYSTEME
;
RET             ;RETOUR AU BASIC
```

Notre petit programme de lancement interdit l'interruption, sauve le contenu de tous les seconds registres sur la pile et appelle votre programme ou votre routine. Les valeurs de départ sont ensuite réécrites dans les seconds registres et l'interruption est à nouveau autorisée, c'est tout.

Comme l'interruption est interdite pendant tout ce temps, le système d'exploitation ne se rend pas compte de l'utilisation des seconds registres. Notez bien que cela implique que votre programme ne doit pas autoriser les interruptions. Si cela s'avérait cependant nécessaire, il faudrait stocker le vecteur d'interruption système à l'adresse &38. Vous pourrez ensuite définir n'importe quel autre vecteur à la place et autoriser à nouveau les interruptions. Peu avant la fin de votre programme, vous devrez enfin restaurer le vecteur initial.

Structure de la RAM vidéo

Le graphisme de l'AMSTRAD CPC permet une résolution maximale de 640 fois 200 points soit 80 fois 200 octets. La RAM vidéo commence normalement à l'adresse &C000 et finit en &FFFF. Le système d'exploitation de l'AMSTRAD ne travaille malheureusement pas avec une organisation interne de la RAM calquée sur la sortie concrète sur l'écran. La structure graphique verticale n'est pas construite ligne de points par ligne de points mais caractère par caractère, c'est-à-dire en sautant chaque fois 7 lignes de points. Le petit programme BASIC que voici vous permettra de mieux le percevoir :

```
10 REM *******************************************
20 REM **                                       **
30 REM ** Remplissage de la mémoire écran       **
30 REM **                                       **
40 REM *******************************************
50 '
55 '
60 MODE 2
70 For I=&C000 to &FFFF
80 POKE I,&FF
90 NEXT
```

Mais pourquoi le système d'exploitation AMSTRAD utilise-t-il une telle organisation de l'écran qui semble très étrange de prime abord? C'est parce qu'elle convient parfaitement pour représenter un caractère de 8 points sur 8. Comme le CPC produit même en mode graphique toutes sortes de textes, cette méthode présente certains avantages sur le plan de la rapidité.

Dans de nombreux autres cas cette méthode se révélera toutefois un obstacle compliqué qui freinera impitoyablement la sortie graphique dans vos programmes Z80.

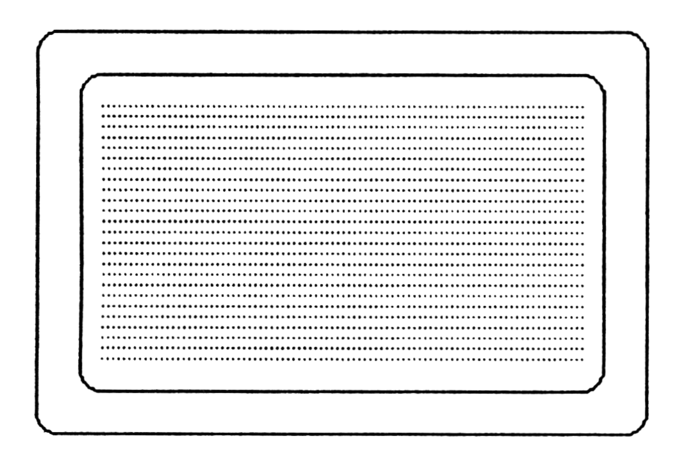


Figure 21 : Structure de l'écran

Bien que l'unité centrale Z80 soit un processeur 8 bits, elle nous offre, comme nous l'avons indiqué au chapitre 9.1, des registres 16 bits ainsi qu'une arithmétique 16 bits. Cette arithmétique concerne l'addition et la soustraction. Cette possibilité va nous permettre de développer un algorithme 16 bits simple pour tracer une ligne verticale.

```
        ; ***********************
        ; **                   **
        ; ** VERTICAL-LINE     **
        ; **                   **
        ; ***********************
        ;
        ;
START:  LD      HL,VSTART       ;adresse de départ sur 16 bits
        LD      DE,&0800        ;décalage de ligne 1
        LD      BC,&3FAF        ;décalage de ligne 2
        LD      A,VCOUNT        ;nombre de points (1-200)
;
```

```
NXTLN:  LD    (HL),COLOR     ;plotter la couleur voulue
        ADD   HL,DE          ;ligne suivante
        JP    NC,NOOFF       ;décalage de ligne 2?
        SBC   HL,BC          ;ligne définitive
;
NOOFF:  DEC   A              ;tous les points ont été plottés?
        RET   Z              ;si oui, terminé, sinon
        JP    NXTLN          ;plotter le point suivant
```

COLOR contient des informations sur l'épaisseur de ligne actuelle et la couleur de ligne qui sont en même temps fonction du mode, c'est-à-dire que chacun des trois modes exige des paramètres spécifiques.

MODE 2 : (620 fois 200 points/2 couleurs)

C'est pour ce mode que les paramètres sont les plus simples à calculer. Chaque bit mis représente un point de la couleur PEN 1, chaque bit annulé la couleur PAPER. Un maximum de 8 lignes peuvent être dessinées simultanément.

MODE 1 : (320 fois 200 points/4 couleurs)

C'est là que les choses se compliquent légèrement. Les couples de bits relatifs (7,3), (6,2), (5,1) et (4,0) représentent chacun un point d'image, un pixel. LD A,1 aura par exemple pour effet d'affecter la couleur PEN 1 à la ligne relative 0.

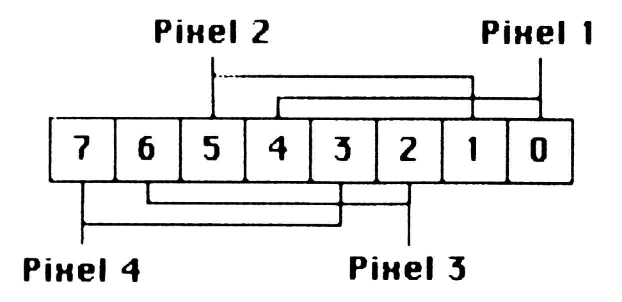


Figure 22 : **Organisation des pixels (MODE 1)**

MODE 0 : (160 x 200 points/16 couleurs)

Ce mode met à votre disposition 4 bits d'informations de couleur. Cela nous permet d'affecter à un point d'image (pixel) 2^4 = 16 couleurs. La couleur PEN du point d'image de gauche est définie par les bits relatifs (7,3,5,1). Les bits relatifs (6,2,4,0) représentent de même les couleurs PEN du point d'image de droite. Lors de la définition des couleurs, il faut absolument tenir compte du fait que la signification des bits ne correspond pas à l'ordre d'un octet binaire habituel. Dans le codage utilisé par AMSTRAD, ce sont en effet les bits 1, 5, 3 et 7 qui correspondent à tous les points de gauche, les bits 0, 4, 2 et 6 codant les couleurs de tous les points de droite.

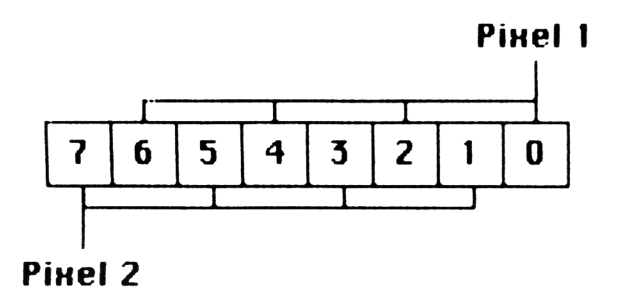
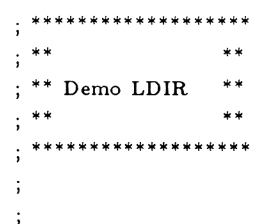


Figure 23 : **Organisation des pixels (MODE 0)**

Il est par contre beaucoup plus facile de tracer une ligne horizontale. Cela peut même être réalisé avec une seule instruction du Z80! Vous ne le croyez pas? Alors étudiez un peu le programme suivant :



```
; ******************
; **              **
; ** Demo LDIR    **
; **              **
; ******************
;
;
```

```
START:  LD    HL,VSTART       ;Début de la RAM vidéo
        LD    DE,VSTART+1     ;2nde valeur pour le transfert
        LD    BC,COUNT        ;longueur de ligne
        LD    A,COLOR         ;couleur de ligne
        LD    (HL),A          ;plotter le premier octet
;
        LDIR                  ;tracer la ligne!
;
        RET                   ;terminé!
```

L'instruction LDIR dessine la ligne horizontale, après l'initialisation, à une vitesse incroyable! Elle ne nécessite que 21 cycles d'horloge par octet. Notez bien que cette opération s'effectue octet par octet.

Le tracé d'une ligne horizontale peut même se faire encore nettement plus vite. Pour cela, nous nous servirons d'un petit truc. Comme l'unité centrale Z80 comporte un pointeur de pile 16 bits, nous pouvons y charger la dernière adresse écran d'une ligne. Si nous chargeons maintenant la valeur de la couleur voulue (en tenant compte du mode) dans les octets fort et faible d'un registre double, deux octets peuvent être plottés simultanément sur l'écran par un simple PUSH. A cet égard, il ne faut jamais oublier que la valeur de départ du pointeur de pile est stockée avant exécution de la routine et qu'il faut restaurer cette valeur une fois l'opération de dessin terminée. Par ailleurs, les interruptions doivent être interdites pendant toute la période d'exécution de la routine.

Cette méthode est sans aucun doute la manière la plus rapide de tracer une ligne horizontale avec le processeur Z80. Le programme machine que voici vous montre comment peut se présenter la réalisation d'une routine de ce type.

```
; *******************************************
; **                                       **
; **  TRACER UNE LIGNE HORIZONTALE         **
; **  AVEC LE POINTEUR DE PILE             **
; **                                       **
; *******************************************
;
;
VEND:    EQU &C050          ;Fin de la ligne
COLOR:   EQU &FFFF          ;Couleur PLOT
COUNT:   EQU 40             ;Plotter 80 octets
;
;
START:   DI                 ;Interdire les interruptions
         LD   (STACK),SP    ;Sauver le pointeur de pile
;
         LD   SP,VEND       ;SP = Fin de la ligne
         LD   HL,COLOR      ;HL = Mot à plotter
         LD   B,COUNT       ;B  = Longueur de la ligne
;
LINE:    PUSH HL            ;Plotter deux octets
         DJNZ LINE          ;Terminé?
;
         LD   SP,(STACK)    ;Aller chercher le pointeur de pile
         EI                 ;Autoriser les interruptions
;
         RET                ;Terminé!
;
STACK:   DEFS 2          ;Place réservée pour le pointeur de pile
```

Organisation de l'écran orientée table

Il reste vrai malgré tout que seul un algorithme assez lent (parce que complexe) peut nous permettre de localiser une coordonnée écran dans la RAM vidéo.

La masse de calculs nécessaires peut cependant être réduite au minimum grâce à une astuce simple, la localisation orientée table. Vous pouvez faire réaliser par un simple programme BASIC une table de toutes les adresses Y. Vous pourrez ensuite POKEr cette table dans n'importe quelle zone de la mémoire.

Votre programme machine n'aura plus alors qu'à lire l'adresse voulue dans votre table et à ajouter la valeur X. Et ce sera tout!

La table Y high/low peut être produite à l'aide du programme BASIC suivant :

```
  1 REM *******************************
  2 REM **                           **
  3 REM ** Generateur de table Y     **
  4 REM **                           **
  5 REM *******************************
  6 '
  7 '
 10 AD=&C000:REM Debut de la video
 20 YH=&6000:REM Table d'octet fort Y
 30 YL=&6100:REM Table d'octet faible Y
 40 '
 50 FOR I=0 to 199
 60 POKE YH+I,INT(AD/256)
 70 POKE YL+I,AD-INT(AD/256)*256
 80 AD=AD+&800
 90 IF AD>-1 THEN AD=AD-&3FB0
100 NEXT
110 '
120 SAVE "YTAB",B,&6000,&1C7
```

Nous avons placé, dans notre exemple, la table des octets forts en &6000 et la table des octets faibles en &6100. Vous pouvez naturellement transférer ces tables dans n'importe quelle autre zone de la mémoire. N'oubliez pas à cet égard que les octets faibles doivent toujours valoir &00. Cela permettra à votre programme machine d'accéder beaucoup plus rapidement aux tables car cela nous permet d'éviter deux additions sur 16 bits. Mais venons-en maintenant à la routine machine proprement dite :

```
; **********************************
; **                              **
; ** Programme de démo TABGET     **
; **                              **
; **********************************
;
;
TBYH:   EQU  &6000          ;Table Y octet fort
TBYL:   EQU  &6100          ;Table Y octet faible
;
        LD   E,27           ;Transmettre X
        LD   A,55           ;Transmettre Y
        CALL CALC           ;Calculer l'adresse
;
; *** C'est ici que vous pouvez intégrer votre
; *** routine personnelle!
;
        RET                 ;Terminé!
;
CALC:   LD   (OFF1+1),A     ;Transmettre Y
        LD   (OFF2+1),A     ;   "
OFF1:   LD   HL,(TBYL)      ;Aller chercher une valeur
OFF2:   LD   A,(TBYH)       ;dans la table
        LD   H,A            ;   "
        SRL  E              ;X=X/4 (MODE 1)
        SRL  E              ;   "
        LD   D,0            ;Ajouter X/4
        ADD  HL,DE          ;   "
        RET                 ;Résultat dans HL
```

Dans cet exemple, l'adresse d'un point a été calculée en MODE 1. La valeur X doit être pour cela divisée par 4. Pour les différents modes, on applique les formules suivantes :

MODE 0 : X = X/2 (160*200 points)
MODE 1 : X = X/4 (320*200 points)
MODE 2 : X = X/8 (640*200 points)

Vous avez peut-être remarqué que le résultat de la division de X est toujours 80. Cela vient du fait qu'une ligne d'écran horizontale est toujours constituée de 80 octets.

En partant de ce fait, nous pouvons maintenant créer sans problème une fonction PLOT dont la vitesse d'exécution sera très supérieure à celle de la routine équivalente du CPC. Elle attendra dans le registre BC la coordonnée X (0 à 319) et dans l'accumulateur la coordonnée Y (0 à 199). On définira la couleur de plot actuelle dans l'adresse COLOR+1. Il faut pour cela charger

```
&00, &0F, &F0 ou &FF
```

dans COLOR+1 avant d'appeler la routine PLOT. Ces valeurs correspondent aux INKs 0 à 3 (Mode 1). Elles manipulent les quartets des masques bits nécessaires par une opération AND.

Notre routine a besoin, en plus de nos tables Y high/low, d'une table de masques bits. Cette table a quatre octets de long et elle sert à réaliser les masques négatifs pour le fond. Nous vous conseillons, pour des raisons de vitesse d'exécution, de placer la table de masques bits, comme les tables Y high/low, à une adresse dont l'octet faible vaut &00.

```
; ***********************
; **                   **
; ** FONCTION PLOT     **
; **                   **
; ***********************
;
; Paramètres          : BC = pos. X (0-319)
; en entrée           : A = pos. Y (0-199)
;                     : (COLOR+1) = couleur de point
;                       (&00, &0F, &F0, &FF)
;
; Registres modifiés : AF, BC, HL
;
;
        ORG  &6200
;
BITNR:  DEFB %01110111          ;masques négatifs
        DEFB %10111011          ;    "
        DEFB %11011101          ;    "
        DEFB %11101110          ;    "
;
;
```

```
;          ORG    &7000          ;adresse de départ
;
TBYH:      EQU    &6000          ;table Y octet faible
TBYL:      EQU    &6100          ;table Y octet fort
BITNR:     EQU    &6200          ;masques bits
;
PLOT:      LD     (OFF1+1),A     ;transmettre Y
           LD     (OFF2+1),A     ;    "
OFF1:      LD     HL,(TBYL)      ;aller chercher valeur
OFF2:      LD     A,(TBYH)       ;dans la table
           LD     H,A            ;    "
           LD     A,&03          ;masquer le numéro
           AND    C              ;de bit
           LD     (OFF3+1),A     ;pointeur sur le masque bits
           SRL    C              ;X=X/4 (MODE 1)
           SRL    C              ;    "
           ADD    HL,BC          ;    "
;
OFF3:      LD     A,(BITNR)      ;aller chercher le masque bits
           LD     C,A            ;    "
           AND    (HL)           ;créer le masque fond
           LD     B,A            ;et le sauvegarder
           LD     A,C            ;aller chercher le masque bits
           CPL                   ;éliminer négatif
COLOR:     AND    &FF            ;colorer
           OR     B              ;et fusionner avec
                                 ;le masque fond
           LD     (HL),A         ;écrire un point dans la
                                 ;RAM vidéo
;
           RET                   ;terminé!
```

Suivant le même principe que pour notre fonction PLOT, nous pouvons réaliser une fonction TEST orientée table. Elle attendra toujours la coordonnée X (0 à 319) dans le registre BC et la coordonnée Y (0 à 199) dans l'accumulateur. Après exécution du programme, l'accumulateur contiendra une des valeurs suivantes :

```
&00, &0F, &F0 ou &FF
```

Ces valeurs indiquent la couleur de point actuelle des coordonnées adressées (INK 0 à 3).

```
; **********************
; **                  **
; ** FONCTION TEST    **
; **                  **
; **********************
;
; Paramètres           : BC = pos. X (0-319)
; en entrée            :  A = pos. Y (0-199)
;
; Param. de sortie     :  A = couleur de point
;                      (&00, &0F, &F0, &FF)
;
; Registres modifiés :   AF, BC, HL
;
;
        ORG  &6200
;
BITNR:  DEFB %01110111          ;masques négatifs
        DEFB %10111011          ;      "
        DEFB %11011101          ;      "
        DEFB %11101110          ;      "
;
;
;       ORG  &7000              ;adresse de départ
;
TBYH:   EQU  &6000              ;table Y octet faible
TBYL:   EQU  &6100              ;table Y octet fort
BITNR:  EQU  &6200              ;masques bits
;
TEST:   LD   (OFF1+1),A         ;transmettre Y
        LD   (OFF2+1),A         ;    "
OFF1:   LD   HL,(TBYL)          ;aller chercher valeur
OFF2:   LD   A,(TBYH)           ;dans la table
        LD   H,A                ;    "
        LD   A;&03              ;masquer le numéro
        AND  C                  ;de bits
        LD   (OFF3+1),A         ;pointeur sur masque bits
        SRL  C                  ;X=X/4 (MODE 1)
```

```
        SRL   C              ;  "
        ADD   HL,BC          ;  "
;
;OFF3:  LD    A,(BITNR)      ;aller chercher le masque bits
        CPL                  ;masquer les bits
        AND   (HL)           ;superflus
;
        CP    &10            ;bit dans MSN (quartet fort) mis?
        JR    C,NXT          ;si non, alors examiner
                             ;quartet suivant
        LD    B,&F0          ;ranger le résultat
                             ;intermédiaire
NXT:    AND   &0F            ;bit dans LSN (quartet faible) mis?
        JR    Z,NXT2         ;si non, alors
                             ;résultat dans A
        LD    A,&0F          ;ranger résultat
                             ;intermédiaire
        OR    B              ;calculer la couleur de point
        LD    B,A            ;et sauver
NXT2:   LD    A,B            ;couleur de point dans l'accu
;
        RET                  ;terminé!
```

Comme vous le voyez, le principe de la programmation orientée table reste toujours le même. Comme de telles fonctions consomment toujours beaucoup de mémoire à cause de leurs tables, il est déconseillé de les utiliser lorsqu'on n'a pas absolument besoin d'une vitesse d'exécution la plus élevée possible. Dans tous les autres cas, vous pouvez parfaitement utiliser les routines de la ROM de votre CPC chaque fois que possible.

9.5 LES PRINCIPALES ROUTINES DE LA ROM

Nous allons vous présenter maintenant quelques-unes des routines graphiques les plus importantes du système d'exploitation. Les routines sont appelées à travers une table de vecteurs placée en RAM. Il est important de ne pas appeler les routines directement car dans ce cas la compatibilité logicielle entre les modèles 464, 664 et 6128 n'est plus garantie et car d'autre part cela peut "planter" le système.

Les descriptions des routines que nous vous présentons ici sont chacune divisées en 6 parties :

1. Nom de routine et vecteur d'appel
2. Adresses d'appel sur tous les CPCs
3. Brève description de la fonction
4. Conditions d'entrée
5. Conditions de sortie
6. Description complète de la fonction

Toutes les routines graphiques font partie soit du bloc Graphics pack, soit du bloc Text pack ou encore du bloc Screen pack du système d'exploitation AMSTRAD (vous trouverez à la suite des descriptions de routines un listing de la ROM entièrement documenté). Par souci de clarté, les routines de la ROM sont d'abord attribuées grossièrement aux différents packs. Un second classement est ensuite effectué à l'intérieur de chaque pack, par ordre croissant d'adresse. Bien que cette méthode puisse sembler un peu compliquée à première vue, elle vous semblera de plus en plus transparente au fur et à mesure que vous apprendrez à l'utiliser. Cette méthode permet en effet d'établir très facilement des références croisées au listing de la ROM.

Mais voici tout d'abord une liste complète de toutes les routines de la ROM décrites, par ordre alphabétique :

ASK CURSOR	(GRA)	&BBC6
CLEAR WINDOW	(GRA)	&BBDB
CONVERT POS	(GRA)	&BD4F
DOT POSITION	(SCR)	&BC1D
EXTENDED INITIALISE	(GRA)	&BD43
FILL	(GRA)	&BD52
FIRST POINT	(GRA)	&BD49
GET CURSOR	(TXT)	&BB78
GET ORIGIN	(GRA)	&BBCC
GET PAPER	(GRA)	&BBE7
GET PEN	(GRA)	&BBE1

GET WINDOW	(TXT)	&BB69
GET W HEIGHT	(GRA)	&BBD8
GET W WIDTH	(GRA)	&BBD5
HORIZONTAL	(SCR)	&BC5F
INITIALISE	(GRA)	&BBBA
INITIALISE	(TXT)	&BB4E
LINE ABSOLUTE	(GRA)	&BBF6
LINE RELATIVE	(GRA)	&BBF9
MOVE ABSOLUT	(GRA)	&BBC0
MOVE RELATIVE	(GRA)	&BBC3
OUTPUT	(TXT)	&BB5A
PIXELS	(SCR)	&BC5C
PLOT ABSOLUTE	(GRA)	&BBEA
PLOT RELATIVE	(GRA)	&BBED
RESET	(GRA)	&BBBD
RESET	(TXT)	&BB51
SET CURSOR	(TXT)	&BB75
SET GRAPHIC	(TXT)	&BB63
SET MASK	(GRA)	&BD4C
SET ORIGIN	(GRA)	&BBC9
SET PAPER	(GRA)	&BBE4
SET PEN	(GRA)	&BBDE
TEST ABSOLUTE	(GRA)	&BBF0
TEST RELATIVE	(GRA)	&BBF3
TRANS SWITCH	(GRA)	&BD46
VERTICAL	(SCR)	&BC62
WIN ENABLE	(TXT)	&BB66
WIN HIGHT	(GRA)	&BBD2
WIN WIDTH	(GRA)	&BBCF
WR CHAR	(GRA)	&BBFC

TXT INITIALISE **&BB4E**

Adresses de la ROM :	464	&1078
	664	&1070
	6128	&1074

Fonction : Initialiser le pack TXT

Registres d'entrée : aucun

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

TXT INITIALISE est chargé de l'initialisation du pack TXT. Les couleurs PAPER et PEN sont fixées sur leurs valeurs initiales. Toutes les indirections sont copiées dans la RAM et les blocs de paramètres (de 14 octets) de toutes les fenêtres sont initialisés. Comme la routine utilise, pour l'opération de copie, l'instruction de transfert de bloc LDIR de l'unité centrale Z80, tous les doubles registres sont modifiés.

TXT RESET **&BB51**

Adresses de la ROM :	464	&1088
	664	&1080
	6128	&1084

Fonction : Réinitialiser le pack TXT

Registres d'entrée : aucun

Registres de sortie : aucun

Registres modifiés : *AF, BC, DE, HL*

Description de la fonction :

TXT RESET copie l'ensemble de la table de saut des caractères de commande ainsi que les indirections du pack TXT dans la zone RAM appropriée. Comme la routine utilise, pour l'opération de copie, l'instruction de transfert de bloc LDIR de l'unité centrale Z80, tous les doubles registres sont modifiés.

TXT OUTPUT **&BB5A**

Adresses de la ROM :	464	&1400
	664	&13FA
	6128	&13FE

Fonction : Sortir un caractère sur l'écran

Registres d'entrée : A = code ASCII du caractère

Registres de sortie : aucun

Registres modifiés : aucun

Description de la fonction :

TXT OUTPUT envoie sur la fenêtre d'écran actuelle le caractère placé dans l'accumulateur ou bien l'exécute s'il s'agit d'un caractère d'un commande. Notez bien que cette fonction utilise l'indirection TXT OUT ACTION. Si vous avez manipulé cette routine, TXT OUTPUT utilisera également votre routine au lieu de celle figurant dans la ROM.

TXT SET GRAPHIC **&BB63**

Adresses de la ROM :	464	&13A7
	664	&13A4
	6128	&13A8

Fonction : Activer ou désactiver la sortie de texte dans la fenêtre graphique

Registres d'entrée : A = 0 - désactivé, <> 0 - activé

Registres de sortie : aucun

Registres modifiés : AF

Description de la fonction :

TXT SET GRAPHIC permet la sortie de texte dans la fenêtre graphique. Si ce n'est pas un zéro qui est transmis à travers l'accumulateur lors de l'appel de la fonction, le texte apparaîtra désormais à partir de la position actuelle du curseur graphique.

TXT WIN ENABLE **&BB66**

Adresses de la ROM :	464	&120C
	664	&1204
	6128	&1208

Fonction : Définir la taille de la fenêtre de texte actuelle

Registres d'entrée :
D= limite droite de la fenêtre
E = limite inférieure de la fenêtre
H= limite gauche de la fenêtre
L = limite supérieure de la fenêtre

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

TXT WIN ENABLE fixe la taille de la fenêtre de texte actuelle. Les doubles registres HL et DE doivent recevoir les valeurs fixant la taille de la fenêtre avant l'appel de la routine.

TXT GET WINDOW **&BB69**

Adresses de la ROM :	464	&1256
	664	&124E
	6128	&1252

Fonction : Aller chercher la taille de la fenêtre actuelle

Registres d'entrée : aucun

Registres de sortie :
D= limite droite de la fenêtre
E = limite inférieure de la fenêtre
H= limite gauche de la fenêtre
L = limite supérieure de la fenêtre

Registres modifiés : AF

Description de la fonction :

TXT GET WINDOW indique la taille de la fenêtre de texte actuelle. Après abandon de la routine, les registres doubles HL et DE contiennent les valeurs correspondant à la taille de la fenêtre.

TXT SET CURSOR **&BB75**

Adresses de la ROM :	464	&1174
	664	&116C
	6128	&1170

Fonction : Positionner le curseur

Registres d'entrée :
H= colonne
L = ligne

Registres de sortie : aucun

Registres modifiés : AF, HL

Description de la fonction :

TXT SET CURSOR positionne le curseur dans la fenêtre de texte actuelle. Avant appel de la routine, il faut charger la valeur pour la colonne dans H et la valeur pour la ligne dans L.

TXT GET CURSOR **&BB78**

Adresses de la ROM :	464	&1180
	664	&1178
	6128	&117C

Fonction : Déterminer la position actuelle du curseur

Registres d'entrée : aucun

Registres de sortie : A= ROLL-COUNT actuel
H= colonne
L = ligne

Registres modifiés : aucun

Description de la fonction :

TXT GET CURSOR détermine la position actuelle du curseur. Après abandon de la routine, le registre H contient la valeur pour la colonne et L contient la valeur pour la ligne. L'accumulateur contient lui le compteur de scrolling.

SCR DOT POSITION **&BC1D**

Adresses de la ROM :	464	&0BA9
	664	&0BAB
	6128	&0BAF

Fonction : Tester le masque de points

Registres d'entrée :	DE	= Coordonnée X (en fonction du MODE)
	HL	= Coordonnée Y (0 à 199)
Registres de sortie :	B	= nombre de points
	C	= masque de points
	HL	= adresse dans la mémoire écran
Registres modifiés :	AF, DE	

Description de la fonction :

SCR DOT POSITION est une routine qui fournit des informations importantes sur un octet de la mémoire écran. Ces informations servent à la construction orientée masque de l'écran. Elles permettent d'éviter que le fond soit modifié inopinément. SCR DOT POSITION travaille en fonction du MODE et elle attend de ce fait une coordonnée X ne sortant pas du cadre correspondant à chaque mode [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)].

SCR PIXELS (Force Mode) **&BC5C**

Adresses de la ROM :	464	&0C6B
	664	&0C70
	6128	&0C74

Fonction :	Fixer un point	
Registres d'entrée :	B	= numéro de couleur
	C	= masque de points
	HL	= adresse dans la mémoire écran
Registres de sortie :	aucun	
Registres modifiés :	AF	

Description de la fonction :

SCR PIXEL fixe un ou plusieurs points dans la mémoire écran, avec la couleur indiquée dans le registre B. Le mode d'écriture activé n'est pas pris en compte pour cette opération.

SCR HORIZONTAL **&BC5F**

Adresses de la ROM :	464	&0FC4
	664	&0F8F
	6128	&0F93

Fonction : Tracer une ligne horizontale

Registres d'entrée :
A = code couleur
BC = position finale X (suivant le MODE)
DE = position initiale X (suivant le MODE)
HL = position Y (0 à 199)

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

SCR HORIZONTAL trace à grande vitesse une ligne horizontale (de DE à BC) sur l'écran. La coordonnée finale en BC doit toujours être supérieure ou égale à la coordonnée initiale en DE. Tous les paramètres X dépendent du MODE [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)].

SCR VERTICAL **&BC62**

Adresses de la ROM :	464	&102F
	664	&0F97
	6128	&0F9B

Fonction : Tracer une ligne verticale

Registres d'entrée :
A = code couleur
BC = position finale Y (0 à 199)
DE = position X (suivant le MODE)
HL = position initiale Y (0 à 199)

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

SCR VERTICAL trace à grande vitesse une ligne verticale (de DE à BC) sur l'écran. La coordonnée finale en BC doit toujours être supérieure ou égale à la coordonnée initiale en DE. Tous les paramètres X dépendent du MODE [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)].

GRA INITIALISE **&BBBA**

Adresses de la ROM :

464	&15B0
664	&15A4
6128	&15A8

Fonction : Initialisation du pack GRA

Registres d'entrée : aucun

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

Les indirections GRA PLOT, GRA TEST et GRA LINE sont tout d'abord copiées dans la RAM. Les couleurs du fond (PAPER) et du crayon (PEN) sont ensuite fixées sur leurs valeurs par défaut.

ORIGIN est placé dans le coin inférieur gauche de l'écran (0,0). Le curseur graphique est positionné sur 0,0. WINDOW(0) est définie et son contenu est protégé.

GRA RESET **&BBBD**

Adresses de la ROM :	464	&15DF
	664	&15D3
	6128	&15D7

Fonction : Copier toutes les indirections

Registres d'entrée : aucun

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

Les indirections GRA PLOT, GRA TEST et GRA LINE sont copiées dans la RAM.

GRA MOVE ABSOLUTE **&BBC0**

Adresses de la ROM :	464	&15F4
	664	&15FA
	6128	&15FE

Fonction : Positionnement absolu du curseur graphique

Registres d'entrée : DE = position du curseur (colonnes)
HL = position du curseur (lignes)

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

Le curseur graphique est placé dans la position absolue de l'écran définie par DE et HL.

GRA MOVE RELATIVE **&BBC3**

Adresses de la ROM :	464	&15F1
	664	&15F7
	6128	&15FB

Fonction : Positionnement relatif du curseur graphique, par rapport à sa position actuelle

Registres d'entrée : DE = position du curseur (colonnes)
HL = position du curseur (lignes)

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

Le curseur graphique est fixé dans la position relative de l'écran définie par DE et HL. Cette position est calculée en fonction des coordonnées actuelles du curseur graphique. Les coordonnées ainsi obtenues peuvent également être situées en dehors de la fenêtre graphique définie.

GRA ASK CURSOR **&BBC6**

Adresses de la ROM :	464	&15FC
	664	&1602
	6128	&1606

Fonction : Détermine la position actuelle du curseur graphique

---

Registres d'entrée : aucun

Registres de sortie : DE = position du curseur (colonne)
HL = position du curseur (ligne)

Registres modifiés : aucun

Description de la fonction :

Détermine la position actuelle du curseur graphique.

GRA SET ORIGIN **&BBC9**

Adresses de la ROM :	464	&1604
	664	&160A
	6128	&160E

Fonction : Définir l'origine des coordonnées

Registres d'entrée : DE = Coordonnée X
HL = Coordonnée Y

Registres de sortie : aucun

Registres modifiés : DE, HL

Description de la fonction :

L'origine des coordonnées définie par l'utilisateur se réfère toujours au coin inférieur gauche de l'écran.

GRA GET ORIGIN **&BBCC**

Adresses de la ROM :	464	&1612
	664	&1618
	6128	&161C

Fonction : Déterminer l'origine des coordonnées

Registres d'entrée : aucun

Registres de sortie : DE = Coordonnée X
HL = Coordonnée Y

Registres modifiés : aucun

Description de la fonction :

Cette routine détermine l'origine des coordonnées par rapport au coin inférieur gauche de l'écran, comme GRA SET ORIGIN.

GRA WIN WIDTH **&BBCF**

Adresses de la ROM :

464	&1734
664	&16A1
6128	&16A5

Fonction : Fixer les limites gauche et droite de la fenêtre graphique.

Registres d'entrée : DE = bord gauche de la fenêtre
HL = bord droit de la fenêtre

Registres de sortie : aucun

Registres modifiés : AF, DE, HL

Description de la fonction :

GRA WIN WIDTH teste d'abord laquelle des coordonnées transmises à travers DE et HL est la plus grande. Si la coordonnée en HL est la plus petite, les contenus des registres sont interchangés. Ils sont ensuite arrondis pour qu'il y ait toujours un octet entier dans la fenêtre définie par l'utilisateur. Les coordonnées subissent ensuite les manipulations suivantes en fonction du mode écran (MODE 0 à 2) :

MODE 2 : Les coordonnées ne sont pas modifiées.
MODE 1 : Toutes les coordonnées sont divisées par 2
MODE 0 : Toutes les coordonnées sont divisées par 4

GRA WIN HEIGHT **&BBD2**

Adresses de la ROM :	464	&1779
	664	&16E6
	6128	&16EA

Fonction : Fixer les limites supérieure et inférieure de la fenêtre graphique.

Registres d'entrée : DE = limite supérieure de la fenêtre
HL = limite inférieure de la fenêtre

Registres de sortie : aucun

Registres modifiés : AF, DE, HL

Description de la fonction :

GRA WIN HEIGHT teste d'abord laquelle des coordonnées transmises à travers DE et HL est la plus grande. Si la coordonnée en DE est la plus petite, les contenus des registres sont interchangés.

GRA GET W WIDTH **&BBD5**

Adresses de la ROM :	464	&17A6
	664	&1713
	6128	&1717

Fonction : Déterminer les limites gauche et droite de la fenêtre graphique.

Registres d'entrée : aucun

Registres de sortie : DE = limite gauche de la fenêtre
HL = limite droite de la fenêtre

Registres modifiés : AF

Description de la fonction :

GRA GET W WIDTH détermine les limites droite et gauche de la fenêtre. Avant leur sortie, les coordonnées sont adaptées comme suit en fonction du mode d'écran actuel :

MODE 2 : Les coordonnées ne sont pas modifiées.
MODE 1 : Toutes les coordonnées sont divisées par 2
MODE 0 : Toutes les coordonnées sont divisées par 4

GRA GET W HEIGHT **&BBD8**

Adresses de la ROM :	464	&17BC
	664	&1729
	6128	&172D

Fonction : Déterminer les limites supérieure et inférieure de la fenêtre graphique.

Registres d'entrée : aucun

Registres de sortie : DE = limite supérieure de la fenêtre
HL = limite inférieure de la fenêtre

Registres modifiés : AF

Description de la fonction :

GRA GET W HEIGHT détermine les limites supérieure et inférieure de la fenêtre. La fonction est en rapport direct avec GRA GET W WIDTH.

GRA CLEAR WINDOW **&BBDB**

Adresses de la ROM :	464	&17C5
	664	&1732
	6128	&1736

Fonction : Effacer une fenêtre graphique.

Registres d'entrée : aucun

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

GRA CLEAR WINDOW vide la fenêtre graphique et positionne le curseur graphique sur l'origine du système de coordonnées.

GRA SET PEN **&BBDE**

Adresses de la ROM :	464	&17F6
	664	&1763
	6128	&1767

Fonction : Définir la couleur d'un crayon de couleur.

Registres d'entrée : A = couleur du crayon

Registres de sortie : aucun

Registres modifiés : AF

Description de la fonction :

Le code couleur indiqué en A est automatiquement adapté au MODE actuel après quoi la définition de couleur est effectuée.

GRA GET PEN **&BBE1**

Adresses de la ROM :	464	&1804
	664	&1771
	6128	&1775

Fonction : Aller chercher la couleur d'un crayon graphique

Registres d'entrée : aucun

Registres de sortie : A = couleur du crayon

Registres modifiés : AF

Description de la fonction :

GRA GET PEN charge la couleur du crayon graphique actuel dans l'accumulateur. Cette opération modifie tous les flags.

GRA SET PAPER **&BBE4**

Adresses de la ROM :	464	&17FD
	664	&176A
	6128	&176E

Fonction : Définir la couleur du fond (PAPER).

Registres d'entrée : A = couleur du fond

Registres de sortie : aucun

Registres modifiés : AF

Description de la fonction :

La couleur du fond (PAPER) est fixée à l'aide de GRA SET PAPER sur le code couleur placé dans l'accumulateur. Cette opération modifie tous les flags.

GRA GET PAPER **&BBE7**

Adresses de la ROM :	464	&180A
	664	&1776
	6128	&177A

Fonction : Lire la couleur du fond (PAPER).

Registres d'entrée : aucun

Registres de sortie : A = couleur du fond

Registres modifiés : AF

Description de la fonction :

La couleur du fond actuelle (PAPER) est chargée dans l'accumulateur à l'aide de GRA GET PAPER. Cette opération modifie tous les flags.

GRA PLOT ABSOLUTE **&BBEA**

Adresses de la ROM :	464	&1813
	664	&177F
	6128	&1783

Fonction : Sortir un point dans des coordonnées absolues de la fenêtre actuelle.

Registres d'entrée : DE = Position X
HL = Position Y

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

GRA PLOT ABSOLUTE est chargée de la sortie d'un point dans le système de coordonnées défini par l'utilisateur. Si une coordonnée sort de la fenêtre actuelle, la fonction est exécutée mais la sortie sur écran sera ignorée.

GRA PLOT RELATIVE **&BBED**

Adresses de la ROM :	464	&1810
	664	&177C
	6128	&1780

Fonction : Sortir un point dans des coordonnées relatives, dans la fenêtre actuelle.

Registres d'entrée : DE = distance relative X
HL = distance relative Y

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

GRA PLOT RELATIVE sort un point dans le système de coordonnées défini par l'utilisateur. Si une coordonnée relative sort de la fenêtre actuelle, la fonction est exécutée mais la sortie sur écran sera ignorée.

GRA TEST ABSOLUTE **&BBF0**

Adresses de la ROM :	464	&1827
	664	&1793
	6128	&1797

Fonction : Lire la couleur d'un point dans des coordonnées absolues, sur la fenêtre actuelle.

Registres d'entrée : DE = Position X
HL = Position Y

Registres de sortie : A = code couleur du point testé

Registres modifiés : BC, DE, HL

Description de la fonction :

GRA PLOT ABSOLUTE teste la couleur d'un point dans le système de coordonnées défini par l'utilisateur et transmet son code couleur à l'accumulateur. Si une coordonnée sort de la fenêtre actuelle, cette fonction est ignorée.

GRA TEST RELATIVE **&BBF3**

Adresses de la ROM :

464	&1824
664	&1790
6128	&1794

Fonction : Lire la couleur d'un point dans des coordonnées relatives, sur la fenêtre actuelle.

Registres d'entrée : DE = décalage relatif X
HL = décalage relatif Y

Registres de sortie : A = code couleur du point testé

Registres modifiés : BC, DE, HL

Description de la fonction :

GRA PLOT RELATIVE teste la couleur d'un point dans le système de coordonnées défini par l'utilisateur et transmet son code couleur à l'accumulateur. Si une coordonnée relative sort de la fenêtre actuelle, cette fonction est ignorée.

GRA LINE ABSOLUTE **&BBF6**

Adresses de la ROM :

464	&1839
664	&17A5
6128	&17A9

Fonction : Trace une ligne de la position actuelle à la position absolue indiquée, dans la fenêtre définie.

Registres d'entrée : DE = Position X
HL = Position Y

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

GRA LINE ABSOLUTE trace une ligne de la position actuelle du curseur graphique à la position absolue indiquée en DE et HL dans le système de coordonnées défini. La fonction est exécutée même si une coordonnée sort de la fenêtre actuelle. La ligne ne sera toutefois visible que dans la fenêtre définie. C'est le crayon graphique actuel qui est utilisé pour tracer la ligne.

GRA LINE RELATIVE **&BBF9**

Adresses de la ROM :

464	&1836
664	&17A2
6128	&17A6

Fonction : Trace une ligne de la position actuelle à la position relative indiquée, dans la fenêtre définie.

Registres d'entrée : DE = décalage relatif X
HL = décalage relatif Y

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

GRA LINE RELATIVE trace une ligne de la position actuelle du curseur graphique à la position relative indiquée en DE et HL dans le système de coordonnées défini. La fonction est exécutée même si une coordonnée sort de la fenêtre actuelle. La ligne ne sera toutefois visible que dans la fenêtre définie. C'est le crayon graphique actuel qui est utilisé pour tracer la ligne.

GRA WR CHAR **&BBFC**

Adresses de la ROM :	464	&1945
	664	&193C
	6128	&1940

Fonction : Sort un caractère sur l'écran

Registres d'entrée : A = caractère (Code ASCII)

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

GRA WR CHAR sort le caractère dont le code ASCII est dans l'accumulateur avec le coin inférieur gauche du caractère aligné sur la position du curseur graphique. Le curseur graphique est ensuite décalé vers la droite à raison de la largeur du caractère. Ce décalage dépend du MODE, il est de 8 (MODE 2), 16 (MODE 1) ou 32 (MODE 0) points. La couleur de caractère est celle du crayon actuel. La couleur du fond est sortie avec le caractère et elle efface le fond sur la surface de la matrice de caractère.

Cette couleur du fond est identique à la couleur PAPER. Si la position du curseur graphique est située en dehors de la fenêtre définie, GRA WR CHAR est ignoré.

GRA EXTENDED INITIALISE **&BD43**

Adresses de la ROM :	464	-
	664	&15E8
	6128	&15EC

Fonction : Initialiser les fonctions graphiques supplémentaires

Registres d'entrée : aucun

Registres de sortie : aucun

Registres modifiés : AF, HL

Description de la fonction :

GRA EXTENDED INITIALISE initialise, comme GRA INITIALISE, toutes les fonctions supplémentaires des CPC 664 et 6128. Cette fonction n'est pas disponible sur le CPC 464.

GRA TRANS SWITCH **&BD46**

Adresses de la ROM :	464	-
	664	&19D1
	6128	&19D5

Fonction : Activation ou désactivation du mode transparent en mode graphique.

Registres d'entrée : A = commutateur

Registres de sortie : aucun

Registres modifiés : aucun

Description de la fonction :

GRA TRANS SWITCH active ou désactive le mode transparent pour la sortie de caractère en mode graphique. Si le fond doit être fusionné avec les caractères, un 0 doit être transmis à travers l'accumulateur. Si ce n'est pas le cas, le contenu de l'accumulateur devra être différent de zéro. Cette fonction n'est pas disponible sur le CPC 464.

GRA FIRST POINT **&BD49**

Adresses de la ROM :

464	-
664	&17B0
6128	&17AC

Fonction : Décide si le premier point d'une ligne doit être plotté.

Registres d'entrée : A = commutateur

Registres de sortie : aucun

Registres modifiés : aucun

Description de la fonction :

GRA FIRST POINT décide si le premier point d'une ligne doit être plotté ou non. Si le premier point ne doit pas être fixé, l'accumulateur doit contenir un zéro lors de l'appel de la fonction. Dans le cas contraire, le contenu de l'accumulateur devra être différent de zéro. Cette fonction n'est pas disponible sur le CPC 464.

GRA SET MASK **&BD4C**

Adresses de la ROM :	464	-
	664	&17A8
	6128	&17AC

Fonction : Définir le masque pour la fonction LINE

Registres d'entrée : A = masque

Registres de sortie : aucun

Registres modifiés : aucun

Description de la fonction :

GRA SET MASK permet de définir l'apparence d'une ligne. Il faut pour cela placer un masque de 8 bits dans l'accumulateur lors de l'appel de la fonction. Chaque bit mis correspondra à un point sur l'écran. Voici quelques exemples :

Tirets	:	%11110000 (&F0)
Pointillé	:	%10101010 (&AA)
Trait-point	:	%11100100 (&E4)
Ligne pleine	:	%11111111 (&FF)

Cette fonction n'est pas disponible sur le CPC 464.

GRA CONVERT POS **&BD4F**

Adresses de la ROM :	464	-
	664	&1626
	6128	&162A

Fonction : Convertir des coordonnées absolues en coordonnées physiques.

Registres d'entrée :
DE = Coordonnée absolue X (0 à 639)
HL = Coordonnée absolue Y (0 à 399)

Registres de sortie :
DE = Coordonnée phys. X (MODE)
HL = Coordonnée phys. Y (0-199)

Registres modifiés : AF

Description de la fonction :

GRA CONVERT POS convertit les coordonnées absolues (d'après l'origine) indiquées dans DE et HL en coordonnées physiques (d'après le MODE). Cela signifie que la valeur Y sera toujours divisée par 2 alors que la valeur X sera divisée suivant le MODE par 4, 2 ou 1 [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)]. Cette fonction n'est pas disponible sur le CPC 464.

GRA FILL **&BD52**

Adresses de la ROM :

464	-
664	&19D5
6128	&19D9

Fonction : Remplir d'une couleur une surface quelconque

Registres d'entrée :

A = couleur de remplissage
DE = taille maximale du buffer FILL
HL = adresse de départ du buffer FILL

Registres de sortie : aucun

Registres modifiés : AF, BC, DE, HL

Description de la fonction :

GRA FILL remplit une surface fermée de n'importe quelle structure avec la couleur indiquée dans l'accumulateur et à partir de la position du curseur graphique. La couleur de la limite de la surface doit coïncider avec la couleur actuelle du crayon ainsi qu'avec la couleur de remplissage. Comme GRA FILL travaille de façon récursive, elle a besoin d'un buffer dont la taille et l'emplacement doivent être indiqués dans les doubles registres DE et HL. Si le buffer défini est trop petit, l'opération de remplissage sera automatiquement interrompue. Cette fonction n'est pas disponible sur le CPC 464.

9.6 LE LISTING DE LA ROM

Nous allons vous présenter dans cette partie de l'ouvrage une explication détaillée des sections GRA, TXT et SCR de la ROM de l'AMSTRAD, c'est-à-dire de toutes les parties ayant trait au graphisme et grâce auxquelles vous pourrez développer plus facilement des programmes graphiques en langage machine. Peut-être savez-vous déjà que les différences entre les ROMs des différents modèles d'ordinateurs AMSTRAD sont heureusement réduites au minimum. C'est pourquoi nous avons délibérément axé notre description sur le CPC 6128 afin de ne pas gâcher trop de papier. Cela ne signifie absolument pas que les possesseurs des autres modèles doivent refermer le livre. Il faudra simplement qu'ils fassent un peu attention et vérifient chaque fois si les commentaires imprimés correspondent bien directement au listing de leur ROM ou si, au contraire, les adresses ne sont pas décalées de quelques octets. Pour vous intéresser à ces problèmes vous devez de toute façon connaître suffisamment le système d'exploitation pour pouvoir faire vous-même quelques expériences et tests. Vous ne devriez donc pas avoir trop de mal à effectuer les ajustements nécessaires. Si vous ne vous sentez toutefois pas assez d'assurance dans ce domaine, nous vous conseillons d'utiliser exclusivement les vecteurs.

Du fait de la législation sur le droit d'auteur, il ne nous est malheureusement pas possible d'imprimer dans cet ouvrage les codes d'opération et les mnémoniques de la ROM AMSTRAD. Nous avons donc été contraints à vous présenter uniquement les commentaires et les adresses du listing. Ces commentaires ne vous aideront bien sûr pas énormément tant que vous ne disposerez pas du code assembleur correspondant. C'est pourquoi nous vous proposons un désassembleur qui vous permettra de produire vous-même un listing de la zone de la ROM que nous voulons examiner. Si vous recoupez ensuite les adresses du listing avec celles indiquées dans les commentaires, vous disposerez d'un commentaire extrêmement précieux du listing de la ROM.

9.6.1 Désassembleur Z80 pour le CPC

L'accès à la mémoire ROM de l'AMSTRAD CPC pose un problème insoluble à partir du niveau du BASIC. Cependant les programmes d'application qui exploitent des données de la ROM, comme par exemple un désassembleur, demandent un gros travail de programmation si on veut les réaliser entièrement en langage machine. C'est pourquoi nous allons vous présenter ici un désassembleur Z80 écrit en BASIC mais qui utilise une petite routine machine pour lire la ROM. Il vous permettra d'abord de mieux comprendre le fonctionnement de l'accès à la ROM mais il constituera également pour vous un outil indispensable pour examiner le système d'exploitation de l'AMSTRAD CPC.

Dès après le lancement du désassembleur, le programme écrit une petite routine machine dans la mémoire du CPC, à partir de l'adresse &AB70. Cette routine se présente ainsi en assembleur :

```
AB70   01 82 7F    LD    BC,&7F82      ;Configuration RAM-ROM
AB73   ED 49       OUT   (C),C         ;Commutation
AB75   1A          LD    A,(DE)        ;Lire un octet dans la ROM
AB76   32 7F AB    LD    (&AB7F),A     ;Et l'écrire dans la RAM
AB79   C9          RET                 ;Terminé!
```

Le désassembleur accédera à ce programme machine chaque fois qu'il voudra lire la ROM. Lors de l'appel de la routine avec CALL, l'adresse à lire est transmise comme paramètre. L'adresse est écrite par le BASIC dans le registre DE du Z80 où elle est disponible pour la routine machine. L'octet lu dans la ROM est écrit par la routine à l'adresse &AB7F de la mémoire RAM où il pourra être lu par la fonction BASIC PEEK. C'est ainsi que les informations figurant dans la ROM peuvent être rendues accessibles au BASIC.

```
100 'COMMUTATION RAM-ROM
101 '
102 DATA &01,&82,&7f,&ed,&49,&1a,&32,&7f,&ab,&c9
103 MEMORY &9FFF
104 FOR a=&AB70 TO &AB79
105 READ d
106 POKE a,d
107 NEXT a
```

```
108 '
109 MODE 2
110 GOTO 227
111 LOCATE 18,4:PRINT"D E S A S S E M B L E U R  Z 8 0"
112 LOCATE 5,7:INPUT"Imprimante (o/n) ",e$
113 IF e$="o" THEN aus=8 ELSE aus=0
114 LOCATE 5,10:INPUT"Adresse de depart : &",a$
115 GOSUB 213:anfa=a
116 LOCATE 5,12:INPUT"Adresse finale    : &",a$
117 GOSUB 213:ende=a
118 IF anfa>ende THEN 109
119 pc=anfa
120 MODE 2
121 adr=pc
122 PRINT#aus,HEX$(adr,4);"  ";
123 iflag=0
124 GOSUB 219
125 GOSUB 137
126 IF iflag THEN 178
127 IF (w=&CF OR w=&D7 OR w=&DF OR w=&EF) AND (LEFT$(pr$,3)="RST")
THEN pr$=pr$+" /DW:nn"
128 IF INSTR(pr$,"n")<>0 THEN 191
129 IF INSTR(pr$,"e")<>0 THEN 203
130 po=INSTR(pr$," ")
131 IF PR$="" THEN PR$="???"
132 IF po=0 THEN PRINT#aus,TAB(21);pr$;:GOTO 134
133 PRINT#aus,TAB(21);LEFT$(pr$,po-1);TAB(27);RIGHT$(pr$,LEN(pr$)-
po);
134 PRINT#aus
135 IF pc<=ende THEN 121
136 END
137 '
139 '
138 'INTERPRETER
140 IF (w=&DD OR w=&FD) AND NOT iflag THEN 163
141 IF w=&ED THEN 158
142 IF w=&CB THEN 150
143 GOSUB 170
144 ON co1 GOTO 146,148,145
145 pr$=bef$(w):RETURN
146 IF w=&76 THEN pr$="HALT":RETURN
147 pr$="LD "+regtab$(co2)+","+reg$:RETURN
```

```
148 IF co2=0 OR co2=1 OR co2=3 THEN a$=" A," ELSE a$=" "
149 pr$=arilog$(co2)+a$+reg$:RETURN
150 '
151 'CB
152 '
153 GOSUB 219
154 IF iflag THEN dis=w:GOSUB 219
155 GOSUB 170
156 IF co1=0 THEN pr$=rotschi$(co2)+" "+reg$ ELSE pr$=bitti$(co1)+S
TR$(co2)+","+reg$
157 RETURN
158 '
159 'ED
160 '
161 GOSUB 219
162 IF w<&40 OR w>&BF THEN pr$="???":RETURN ELSE GOTO 145
163 '
164 'XY
165 '
166 iflag=-1
167 IF w=&DD THEN i$="IX" ELSE i$="IY"
168 GOSUB 219
169 GOTO 137
170 '
171 'DECOMPOSER LE CODE
172 '
173 co1=(w AND &X11000000)/64
174 co2=(w AND &X111000)/8
175 co3=w AND &X111
176 reg$=regtab$(co3)
177 RETURN
178 '
179 'INDIXE
180 '
181 po=INSTR(pr$,"HL")
182 IF po=0 THEN pr$="???":GOTO 130
183 IF INSTR(pr$,"(HL)")<>0 THEN 187
184 IF pr$="EX DE,HL" THEN pr$="???":GOTO 130
185 IF pr$="ADD HL,HL" THEN pr$="ADD "+i$+","+i$:GOTO 130
186 pr$=LEFT$(pr$,po-1)+i$+RIGHT$(pr$,LEN(pr$)-po-1):GOTO 127
187 IF LEFT$(pr$,2)="JP" THEN 186
188 IF pc-adr<3 THEN GOSUB 219:dis=w
```

```
189 IF dis>127 THEN dis$=STR$(dis-
256) ELSE dis$="+"+RIGHT$(STR$(dis),LEN(STR$(dis))-1)
190 i$=i$+dis$:GOTO 186
191 'remplacer n
192 po=INSTR(pr$,"nn")
193 IF po<>0 THEN 198
194 po=INSTR(pr$,"n")
195 GOSUB 219
196 pr$=LEFT$(pr$,po-1)+"&"+HEX$(w,2)+RIGHT$(pr$,LEN(pr$)-po)
197 GOTO 130
198 GOSUB 219:lb=w
199 GOSUB 219
200 wert=w*256+lb
201 pr$=LEFT$(pr$,po-1)+"&"+HEX$(wert,4)+RIGHT$(pr$,LEN(pr$)-po-1)
202 GOTO 130
203 '
204 'REMPLACER E
205 '
206 po=INSTR(pr$,"e")
207 GOSUB 219
208 IF w>127 THEN w=w-256:REM Complement a 2
209 w=w+2
210 a$="$"+STR$(w)+" >"+"&"+HEX$(pc+w-2,4)
211 pr$=LEFT$(pr$,po-1)+a$+RIGHT$(pr$,LEN(pr$)-po)
212 GOTO 130
213 '
214 'CONVERSION HEXA - DECIMAL
215 '
216 IF a$="" THEN a=0:RETURN
217 a=VAL("&"+a$)
218 RETURN
219 '
220 'LIRE UN OCTET DANS LA ROM
221 '
222 CALL &AB70,PC
223 W=PEEK(&AB7F)
224 pc=pc+1
225 PRINT#aus,HEX$(w,2);" ";
226 RETURN
227 '
228 'INIT
229 '
```

```
230 DIM regtab$(7),rotschi$(8),bitti$(3),arilog$(7),bef$(255)
231 FOR i=0 TO 7:READ regtab$(i):NEXT
232 FOR i=0 TO 7:READ rotschi$(i):NEXT
233 FOR i=1 TO 3:READ bitti$(i):NEXT
234 FOR i=0 TO 7:READ arilog$(i):NEXT
235 FOR i=0 TO &7F:READ bef$(i):NEXT
236 FOR i=&80 TO &9F:bef$(i)="":NEXT
237 FOR i=&A0 TO &FF:READ bef$(i):NEXT
238 GOTO 111
239 '
240 'DATAS
241 '
242 DATA B,C,D,E,H,L,(HL),A
243 DATA RLC,RRC,RL,RR,SLA,SRA,???,SRL
244 DATA BIT,RES,SET
245 DATA ADD,ADC,SUB,SBC,AND,XOR,OR,CP
246 DATA NOP,"LD BC,nn","LD (BC),A",INC BC,INC B,DEC B,"LD B,n",RLC
A
247 DATA "EX AF,AF'","ADD HL,BC","LD A,(BC)",DEC BC,INC C,DEC C,"LD
 C,n",RRCA
248 DATA DJNZ e,"LD DE,nn","LD (DE),A",INC DE,INC D,DEC D,"LD D,n",
RLA
249 DATA JR e,"ADD HL,DE","LD A,(DE)",DEC DE,INC E,DEC E,"LD E,n",R
RA
250 DATA "JR NZ,e","LD HL,nn","LD (nn),HL",INC HL,INC H,DEC H,"LD H
,n",DAA
251 DATA "JR Z,e","ADD HL,HL","LD HL,(nn)",DEC HL,INC L,DEC L,"LD L
,n",CPL
252 DATA "JR NC,e","LD SP,nn","LD (nn),A",INC SP,INC (HL),DEC (HL),
"LD (HL),n",SCF
253 DATA "JR C,e","ADD HL,SP","LD A,(nn)",DEC SP,INC A,DEC A,"LD A,
n",CCF
254 DATA "IN B,(C)","OUT (C),B","SBC HL,BC","LD (nn),BC",NEG,RETN,I
M 0,"LD I,A"
255 DATA "IN C,(C)","OUT (C),C","ADC HL,BC","LD BC,(nn)",,RETI,,"LD
 R,A"
256 DATA "IN D,(C)","OUT (C),D","SBC HL,DE","LD (nn),DE",,,IM 1,"LD
 A,I"
257 DATA "IN E,(C)","OUT (C),E","ADC HL,DE","LD DE,(nn)",,,IM 2,"LD
 A,R"
258 DATA "IN H,(C)","OUT (C),H","SBC HL,HL","LD (nn),HL",,,,RRD
259 DATA "IN L,(C)","OUT (C),L","ADC HL,HL","LD HL,(nn)",,,,RLD
```

```
260 DATA ,,"SBC HL,SP","LD (nn),SP",,,,
261 DATA "IN A,(C)","OUT (C),A","ADC HL,SP","LD SP,(nn)",,,,
262 DATA LDI,CPI,INI,OUTI,,,,,LDD,CPD,IND,OUTD,,,,
263 DATA LDIR,CPIR,INIR,OTIR,,,,,LDDR,CPDR,INDR,OTDR,,,,
264 DATA RET NZ,POP BC,"JP NZ,nn",JP nn,"CALL NZ,nn",PUSH BC,"ADD A
,n",RST &00
265 DATA RET Z,RET,"JP Z,nn",-
>,"CALL Z,nn",CALL nn,"ADC A,n",RST &08
266 DATA RET NC,POP DE,"JP NC,nn","OUT (n),A","CALL NC,nn",PUSH DE,
"SUB n",RST &10
267 DATA RET C,EXX,"JP C,nn","IN A,(n)","CALL C,nn",-
>,"SBC A,n",RST &18
268 DATA RET PO,POP HL,"JP PO,nn","EX (SP),HL","CALL PO,nn",PUSH HL
,"AND n",RST &20
269 DATA RET PE,JP (HL),"JP PE,nn","EX DE,HL","CALL PE,nn",-
>,"XOR n",RST &28
270 DATA RET P,POP AF,"JP P,nn",DI,"CALL P,nn",PUSH AF,"OR n",RST &
30
271 DATA RET M,"LD SP,HL","JP M,nn",EI,"CALL M,nn",-
>,"CP n",RST &38
```

9.6.2 Screen Pack (SCR)

Le SCREEN PACK est subordonné au TEXT PACK et au GRAPHICS PACK. Il représente en quelque sorte la force de travail de ces deux packs puisqu'il est chargé de la gestion directe de l'écran.

0ABF SCR INITIALISE

Lors de l'appel de cette routine, une initialisation complète du Screen-Pack est effectuée. Le SCREEN START est fixé sur &C000.

```
0ABF  ;ADRESSE DE BASE DES COULEURS DEFAUT DANS DE
0AC2  ;MC CLEAR INKS
0AC5  ;OCTET FORT VALEUR DE DEPART DE SCREEN START
0AC7  ;(OCTET FORT SCREEN START)
0ACA  ;SCR RESET
0ACD  ;FIXER MODE 1 DE L'ECRAN
```

0AD0 SCR RESET

Réinitialisation du pack Screen.

```
0AD0  ;ANNULER L'ACCU ET REINITIALISER LES FLAGS
0AD1  ;SCR ACCESS
0AD4  ;RESTORE SCR INDIRECTIONS
0AD7  ;MOVE (HL+3) DANS ((HL+1)), CNT=(HL)
0ADA  ;REINITIALISER COULEURS
0ADD  ;9 OCTETS
0ADE  ;ADRESSE DE DESTINATION

0AE0  ;CODE POUR JP
0AE1  ;SCR READ

0AE3  ;CODE POUR JP
0AE4  ;SCR WRITE

0AE6  ;CODE POUR JP
0AE7  ;SCR CLEAR
```

0AE9 SCR SET MODE

SCR SET MODE change le mode de l'écran. Lors de l'appel de cette routine, l'accumulateur doit contenir la valeur correspondant au nouveau mode écran. Les valeurs autorisées sont 0, 1 ou 2.

```
0AE9  ;MASQUER LES BITS 2 A 7
0AEB  ;MODE SUPERIEUR OU EGAL A 3?
0AED  ;SI OUI, ALORS RETOUR
0AEE  ;SAUVER LE MODE ECRAN
0AEF  ;TERMINATE COLOUR EVENT BLOCK
0AF2  ;ALLER CHERCHER LE CONTENU DE DE
0AF3  ;DECODER LES COULEURS
0AF6  ;SAUVER L'ACCU ET LES FLAGS
0AFA  ;ALLER CHERCHER LE CONTENU DU REGISTRE HL
0AFB  ;ALLER CHERCHER LE MODE
0AFC  ;ALLER CHERCHER LES MASQUES
0AFF  ;SCR CLEAR
0B02  ;ALLER CHERCHER PEN ET PAPER
0B06  ;ALLER CHERCHER LE NUMERO DE FENETRE
0B07  ;FIXER LES PARAMETRES DE FENETRE SUR LES VALEURS DEFAUT
0B0A  ;INSERT COLOUR EVENT BLOCK
```

0B0C SCR GET MODE

SCR GET MODE fournit dans l'accumulateur le mode actuel de l'écran. Les flags sont fixés comme suit suivant le mode :

Mode 0 : Flag ZERO=0, Flag CARRY=1
Mode 1 : Flag ZERO=1, Flag CARRY=0
Mode 2 : Flag ZERO=0, Flag CARRY=0

```
0B0C  ;(CURR. SCREEN MODE)
0B0F  ;MANIPULER LES FLAGS
0B11  ;RETOUR
```

0B12 FIXER LE MODE 1

Place l'écran en MODE 1

```
0B12   ;VALEUR POUR CURR. SCREEN MODE
0B14   ;ECRIRE CETTE VALEUR
```

0B17 SCR CLEAR

Vider l'écran.

```
0B17   ;TERMINATE COLOUR EVENT BLOCK
0B1A   ;PARAMETRE A TRANSMETTRE POUR SCR SET OFFSET
0B1D   ;SCR SET OFFSET
0B20   ;(ADRESSE SCREEN START)
0B23   ;FIXER OCTET FAIBLE SUR ZERO
0B25   ;HL=ADRESSE DE BASE
0B26   ;DE=ADRESSE DE BASE+1
0B28   ;16 K
0B2B   ;OCTET FIXE SUR ZERO
0B2C   ;VIDER L'ECRAN
0B2E   ;INSERT COLOUR EVENT BLOCK
0B31   ;(CURR. SCREEN MODE)
0B34   ;MC SET MODE (FIXER LE MODE ECRAN)
```

0B37 SCR SET OFFSET

Fixer l'adresse de départ du premier caractère par rapport à l'adresse de base de la RAM vidéo.

```
0B37   ;(SCR HIGH BYTE SCREEN START)
0B3A   ;C'EST ICI QUE CELA CONTINUE
```

0B3C SCR SET BASE

Adresse de base de la RAM vidéo

```
0B3C   ;(POSITION SUR UNE LIGNE)
0B3F   ;SCR MODIFICATION DE SCREEN START
0B42   ;MC SCREEN OFFSET
```

0B45 SCR MODIFICATION DE SCREEN START

La routine reçoit comme paramètres d'entrée la position sur une ligne, dans le registre HL, et l'octet fort de la banque définie comme mémoire écran, dans l'accumulateur.

```
0B45   ;MASQUER LES BITS 0 A 5
0B47   ;(OCTET FORT SCREEN START)
0B4A   ;SAUVER L'ACCU ET LES FLAGS
0B4B   ;CHARGER LE CONTENU DE H DANS L'ACCUMULATEUR
0B4C   ;MASQUER LES BITS 3 A 7
0B4E   ;RENVOYER LE RESULTAT DANS H
0B4F   ;ANNULER LE BIT 0
0B51   ;(POSITION SUR UNE LIGNE)
0B54   ;RETABLIR ANCIEN ETAT ACCU
0B55   ;ET FLAGS
```

0B56 0B50 SCR GET LOCATION

Après abandon de cette routine, l'accumulateur contient l'octet fort du début de l'écran (screen start) et le registre HL contient la position sur une ligne.

```
0B56   ;(POSITION SUR UNE LIGNE)
0B59   ;OCTET FORT SCREEN START
0B5C   ;RETOUR
```

0B5D SCR CHAR LIMITS

SCR CHAR LIMITS fournit le nombre maximum de lignes et de colonnes en fonction du mode écran (MODE). Le nombre de lignes possibles dans le mode actuel est fourni dans le registre C, celui des colonnes dans le registre B.

```
0B5D   ;SCR GET MODE
0B60   ;B: NOMBRE DE COLONNES MODE 0
       ;C: NOMBRE DE LIGNES
       ;(POSITION 0,0 EST PERMISE!)
0B63   ;MODE 0 EST FIXE
0B64   ;NOMBRE DE COLONNES MODE 1
```

```
0B66   ;MODE 1 EST FIXE
0B67   ;NOMBRE DE COLONNES MODE 2
0B69   ;RETOUR
```

0B6A SCR CHAR POSITION

Convertir les coordonnées physiques en une position sur l'écran. SCR CHAR POSITION attend dans le double registre HL, comme paramètres, les coordonnées de colonne (H) et de ligne (L). En sortie, l'adresse de la position de caractère se trouvera dans le registre HL et la longueur d'un caractère dans le sens des X dans le registre B.

```
0B6A   ;SAUVER CONTENU DE DE SUR LA PILE
0B6B   ;SCR GET MODE
0B6E   ;LONGUEUR DE CARACTERE = 4 OCTETS (MODE 0)
0B70   ;MODE 0 ACTIVE?
0B72   ;LONGUEUR DE CARACTERE = DEUX OCTETS (MODE 1)
0B74   ;MODE 1 ACTIVE?
0B76   ;LONGUEUR DE CARACTERE = UN OCTET (MODE 2)
0B77   ;SAUVER L'EXTENSION X DES CARACTERES
0B78   ;ALLER CHERCHER COLONNE
0B79   ;ANNULER OCTET FORT DE COLONNE
0B7B   ;REGISTRE H PASSE A ZERO
0B7C   ;SAUVER COLONNE SUR LA PILE
0B7D   ;REGISTRE D OCTET FORT LIGNE
0B7E   ;REGISTRE E OCTET FAIBLE LIGNE
0B7F   ;HL=HL*80
0B80   ;VOIR 0B7F
0B81   ;VOIR 0B7F
0B82   ;VOIR 0B7F
0B83   ;VOIR 0B7F
0B84   ;VOIR 0B7F
0B85   ;VOIR 0B7F
0B86   ;ALLER CHERCHER COLONNE
0B87   ;AJOUTER COLONNE
0B88   ;SUIVANT MODE, AJOUTER COLONNE, COLONNE*2 OU
       ;COLONNE*4 A HL
0B8A   ;ALLER CHERCHER DECALAGE D'ECRAN
0B8E   ;AJOUTER DECALAGE ECRAN
0B8F   ;OCTET FORT DANS ACCUMULATEUR
```

```
0B90   ;MASQUER LES BITS 3 A 7
0B92   ;REECRIRE LE RESULTAT
0B93   ;(OCTET FORT SCREEN START)
0B96   ;ADRESSE DE L'ECRAN + OCTET FORT SCREEN START
0B97   ;RESULTAT DANS REGISTRE H
0B98   ;ALLER CHERCHER EXTENSION X DU CARACTERE
0B99   ;RESTAURER L'ANCIEN CONTENU DE DE
0B9A   ;TERMINE!
```

0B9B COMPUTE WINDOW PARAMS

La routine attend, comme paramètres d'entrée, la limite gauche de la fenêtre dans le registre H et la limite supérieure dans le registre L. Le registre D devra de même contenir la limite droite de la fenêtre et le registre E la limite inférieure de la fenêtre. Lors de la sortie, l'adresse supérieure gauche de la fenêtre sur l'écran se trouve dans le registre HL et le registre D indique le nombre d'octets par ligne de la fenêtre. Le registre E contient le nombre de lignes de grille de la fenêtre.

```
0B9B   ;LIMITE INFERIEURE DE LA FENETRE DANS L'ACCUMULATEUR
0B9C   ;CALCULER LA HAUTEUR DE LA FENETRE
0B9D   ;AUGMENTER DE UN LA HAUTEUR DE LA FENETRE
0B9E   ;RESULTAT FOIS HUIT
0B9F   ;DONNE LE NOMBRE
0BA0   ;DE LIGNES DE GRILLE DE LA FENETRE
0BA1   ;NOMBRE DE LIGNES DE GRILLE DANS LE REGISTRE E
0BA2   ;LIMITE DROITE DE LA FENETRE DANS L'ACCUMULATEUR
0BA3   ;CALCULER LA LARGEUR DE LA FENETRE
0BA4   ;AUGMENTER LE RESULTAT DE UN
0BA5   ;REECRIRE DANS LE REGISTRE D
0BA6   ;SCR CHAR POSITION
0BA9   ;ANNULER ACCU ET REINITIALISER LES FLAGS
0BAA   ;SUIVANT LE MODE D
0BAB   ;MULTIPLIER PAR 2, 4 OU 8
0BAD   ;RENVOYER LE RESULTAT DANS LE REGISTRE D
0BAE   ;RETOUR
```

0BAF SCR DOT POSITION

Calculer la position sur l'écran d'un point. Lors de l'appel, SCR DOT POSITION attend dans le registre HL la coordonnée X et dans le registre DE la coordonnée Y. Lors de l'abandon de la routine, celle-ci fournit l'adresse écran du point dans le registre HL.

```
0BAF  ;SAUVER LA COORDONNEE X SUR LA PILE
0BB0  ;COORDONNEE Y DANS DE
0BB1  ;VALEUR MAXI POUR COORDONNEE Y
0BB4  ;ANNULER LE CARRY POUR SBC
0BB5  ;MAXIMUM MOINS COORDONNEE Y
0BB7  ;OCTET FAIBLE DU RESULTAT DANS ACCU
0BB8  ;LIBERER ADRESSE DE GRILLE
0BBA  ;TRIPLE ROTATION
0BBB  ;DE TOUS LES BITS SUR LA GAUCHE
0BBC  ;(INTIALISATION ADRESSE DE GRILLE POUR GATE-ARRAY)
0BBD  ;RESULTAT DANS REGISTRE C
0BBE  ;OCTET FAIBLE DE COORDONNEE Y MODIFIEE DANS A
0BBF  ;MASQUER LES BITS 0 A 2
0BC1  ;RENVOYER LE RESULTAT DANS LE REGISTRE L
0BC2  ;OCTET FORT MODIFIE DE LA COORDONNEE Y
0BC3  ;DE = COORDONNEE Y MODIFIEE
0BC4  ;MULTIPLIER COORDONNEE Y
0BC5  ;PAR 10
0BC6  ;ET
0BC7  ;CALCULER LIGNE SO
0BC8  ;ALLER CHERCHER COORDONNEE X SUR LA PILE
0BC9  ;LIBERER BC
0BCA  ;GET PIXEL MASK
0BCD  ;ACCU = MASQUE POINTS
0BCE  ;CALC MASQUE Y-INDEX
0BCF  ;RESULTAT NUL?
0BD1  ;DIVISER INDEX X PAR DEUX
0BD3  ;MASQUE-INDEX-1
0BD4  ;RESULTAT NUL?
0BD6  ;ECHANGE OCTETS FORT ET FAIBLE AVEC PILE
0BD7  ;VOIR 0BD6
0BD8  ;VOIR 0BD6
0BD9  ;VOIR 0BD6
0BDA  ;ALLER CHERCHER MASQUE POINTS
0BDB  ;ET DECALER POINTS DANS LE MASQUE
```

```
0BDC   ;POS X = POS X/2 (OCTET FORT)
0BDE   ;POS X = POS X/2 (OCTET FAIBLE)
0BE0   ;ET DECALER POINTS DANS LE MASQUE
0BE1   ;BIT FAIBLE DANS CARRY?
0BE3   ;CALCULER ADRESSE ECRAN
0BE4   ;(POSITION SUR UNE LIGNE)
0BE8   ;AJOUTER DEBUT VIDEO
0BE9   ;ALLER CHERCHER OCTET FORT D'ADRESSE ECRAN
0BEA   ;ET LE MODIFIER AVEC ADRESSE DE GRILLE
       ;ACTUELLE
0BEC   ;REECRIRE RESULTAT
0BED   ;(OCTET FORT SCREEN START)
0BF0   ;A = OCTET FORT DE L'ADRESSE ECRAN
0BF1   ;AJOUTER DECALAGE ECRAN
0BF2   ;ET ECRIRE RESULTAT EN RETOUR
0BF3   ;ADRESSE DE DEBUT VIDEO
0BF4   ;OCTET FORT DANS C
0BF5   ;C'EST TOUT!
```

0BF6 GET PIXEL MASK

```
0BF6   ;SCR GET MODE
0BF9   ;MASQUE POINTS POUR MODE 0
0BFC   ;MODE 0 ACTIVE?
0BFD   ;MASQUE POINTS POUR MODE 1
0C00   ;MODE 1 ACTIVE?
0C01   ;MASQUE POINTS POUR MODE 2
0C04   ;MODE 2 ACTIVE!
```

0C05 SCR NEXT BYTE

Fournit dans HL l'adresse écran de la prochaine position d'octet si vous avez placé l'ancienne adresse dans HL avant d'appeler la routine. Cela peut sembler inutile mais c'est en réalité très pratique. Il n'est en effet pas aisé, du fait de l'organisation de l'écran axée sur le mode texte, de déterminer la position d'octet. Par ailleurs la distance dépend du mode.

Notez bien que l'adresse renvoyée n'a aucune signification si la position suivante sort des limites de l'écran.

Elle se situera en effet, dans ce cas, dans la zone des derniers octets de la RAM vidéo, octets qui ne sont pas utilisés pour l'affichage.

```
0C05   ;OCTET FAIBLE DE L'ADRESSE ECRAN +1
0C06   ;RETOUR SI PAS DE DEBORDEMENT
0C07   ;OCTET FORT DE L'ADRESSE ECRAN +1
0C08   ;NOUVEL OCTET FORT DANS ACCU
0C09   ;FIXER LES BITS 3 A 7 SUR ZERO
0C0B   ;RETOUR SI ACCU <> 0
0C0C   ;PLACER OCTET FORT DANS L'ACCU
       ;POUR AUTRES CALCULS
0C0D   ;SOUSTRAIRE &08 DE L'ACCU
0C0F   ;NOUVELLE VALEUR CALCULEE POUR OCTET FORT
0C10   ;RETOUR AVEC ADRESSE ECRAN POUR PROCHAINE
       ;POSITION D'OCTET DANS HL
```

0C11 SCR PREV BYTE

Renvoie dans HL l'adresse écran de la position d'octet précédente si vous avez chargé l'ancienne adresse dans HL avant l'appel de la routine. Comparez avec SCR NEXT BYTE.

```
0C11   ;OCTET FAIBLE D'ADRESSE ECRAN DANS ACCU
0C12   ;OCTET FAIBLE-1
0C13   ;ACCUMULATEUR EGALE 0?
0C14   ;RETOUR SI ACCU <> 0
0C15   ;OCTET FORT ADRESSE ECRAN DANS ACCU
0C16   ;OCTET FORT-1
0C17   ;FIXER BITS 3 A 7 SUR ZERO
0C19   ;RETOUR SI ACCU <> 0
0C1A   ;OCTET FORT DANS L'ACCU POUR
       ;CALCULS ULTERIEURS
0C1B   ;AJOUTER &08 A L'ACCU
0C1D   ;NOUVELLE VALEUR CALCULEE POUR L'OCTET FORT
0C1E   ;RETOUR AVEC ADRESSE ECRAN POUR POSITION D'OCTET
       ;PRECEDENTE DANS HL
```

0C1F SCR NEXT LINE

Fonctionne de façon semblable à SCR NEXT BYTE, si ce n'est que l'adresse écran est calculée en avançant d'une ligne entière. Ici aussi, l'adresse renvoyée est sans valeur si elle sort de la zone représentable sur l'écran.

```
0C1F   ;OCTET FORT DE L'ADRESSE ECRAN DANS ACCU
0C20   ;AJOUTER &08 A L'ACCU
0C22   ;RENVOYER RESULTAT DANS H
0C23   ;DECONNECTER LES BITS 0 A 2 ET 6 A 7
0C25   ;RETOUR DANS CES CONDITIONS
0C26   ;RAMENER H DANS L'ACCU
0C27   ;SOUSTRAIRE &40 DE L'ACCU
0C29   ;RENVOYER RESULTAT DANS H
0C2A   ;OCTET FAIBLE ADRESSE ECRAN DANS ACCU
0C2B   ;AJOUTER &50 A L'ACCU
0C2D   ;RENVOYER RESULTAT DANS L
0C2E   ;RETOUR SI CONDITION REMPLIE
0C2F   ;H+1
0C30   ;RENVOYER H DANS L'ACCU
0C31   ;FIXER BITS 3 A 7 SUR ZERO
0C33   ;RETOUR SOUS CETTE CONDITION
0C34   ;REPLACER H DANS L'ACCU
0C35   ;SOUSTRAIRE &08 DE L'ACCU
0C37   ;RENVOYER RESULTAT DANS H
0C38   ;RETOUR INCONDITIONNEL
```

0C39 SCR PREV BYTE

Fonctionne de façon semblable à SCR PREV BYTE, si ce n'est que l'adresse écran est calculée en reculant d'une ligne entière. Comparez avec SCR NEXT LINE et SCR PREV BYTE.

```
0C39   ;OCTET FORT ADRESSE ECRAN DANS ACCU
0C3A   ;SOUSTRAIRE &08 DE L'ACCU
0C3C   ;RAMENER RESULTAT DANS H
0C3D   ;DECONNECTER LES BITS 0 A 2 ET 6 A 7
0C3F   ;BITS 3 A 5 MIS?
0C41   ;RETOUR A CETTE CONDITION
0C42   ;RAMENER H DANS L'ACCU
```

```
0C43   ;AJOUTER &40 A L'ACCU
0C45   ;REPLACER RESULTAT DANS H
0C46   ;OCTET FAIBLE ADRESSE ECRAN DANS ACCU
0C47   ;SOUSTRAIRE &50 DE L'ACCU
0C49   ;RENVOYER RESULTAT DANS L
0C4A   ;RETOUR A CETTE CONDITION
0C4B   ;REPLACER H DANS L'ACCU
0C4C   ;H-1
0C4D   ;FIXER LES BITS 3 A 7 SUR ZERO
0C4F   ;RETOUR SI CONDITION REMPLIE
0C50   ;RAMENER H DANS L'ACCU
0C51   ;AJOUTER &08 A L'ACCU
0C53   ;RENVOYER RESULTAT DANS H
0C54   ;RETOUR INCONDITIONNEL
```

0C55 SCR ACCESS

Caractères de commande visibles/invisibles. SCR ACCESS attend en entrée le mode écran, dans l'accumulateur.

```
0C55   ;DECONNECTER LES BITS INUTILES
0C57   ;SCR PIXELS (FORCE MODE)
0C5A   ;FORCE MODE?
0C5C   ;MODE XOR DEFINI?
0C5E   ;OCTET FAIBLE MODE XOR
0C60   ;SAUTER SI MODE XOR
0C62   ;OCTET FAIBLE MODE AND
0C64   ;SAUTER SI MODE AND
0C66   ;OCTET FAIBLE MODE OR
0C68   ;CODE D'OPERATION POUR JP
0C6A   ;RENVOYER LE CODE D'OPERATION
0C6D   ;(SCR WRITE INDIRECTION)
0C70   ;RETOUR
```

0C71 SCR WRITE

```
0C71   ;SCR WRITE INDIRECTION
```

0C74 SCR PIXELS (FORCE MODE)

Lors de l'appel de cette routine, le registre B doit contenir l'octet Pen et le registre D le masque bits pour le point. Le registre HL contient l'adresse du point dans la mémoire écran.

```
0C74   ;OCTET PEN DANS ACCUMULATEUR
0C75   ;OPERATION XOR AVEC MEMOIRE ECRAN
0C76   ;MASQUE BITS POUR LE POINT
0C77   ;OPERATION XOR AVEC MEMOIRE ECRAN
0C78   ;ECRIRE DANS LA MEMOIRE ECRAN
0C79   ;IL N'Y A PLUS RIEN A FAIRE
```

0C7A MODE XOR

Lors de l'appel de cette routine, le registre B doit contenir l'octet Pen et le registre D le masque bits pour le point. Le registre HL contient l'adresse du point dans la mémoire écran.

```
0C7A   ;OCTET PEN DANS ACCUMULATEUR
0C7B   ;MASQUE BITS POUR LE POINT
0C7C   ;OPERATION XOR AVEC MEMOIRE ECRAN
0C7D   ;ECRIRE DANS LA MEMOIRE ECRAN
0C7E   ;C'EST TOUT
```

0C7F MODE AND

Lors de l'appel de cette routine, le registre B doit contenir l'octet Pen et le registre D le masque bits pour le point. Le registre HL contient l'adresse du point dans la mémoire écran.

```
0C7F   ;MASQUE BITS POUR POINT DANS ACCU
0C80   ;REALISER MASQUE INVERSE
0C81   ;OPERATION OR AVEC OCTET PEN
0C82   ;ET OPERATION AND AVEC MEMOIRE ECRAN
0C83   ;ECRIRE RESULTAT DANS LA MEMOIRE ECRAN
0C84   ;TERMINE
```

0C85 MODE OR

Lors de l'appel de cette routine, le registre B doit contenir l'octet Pen et le registre D le masque bits pour le point. Le registre HL contient l'adresse du point dans la mémoire écran.

```
0C85   ;OCTET PEN DANS ACCU
0C86   ;OPERATION AND AVEC MASQUE BITS
0C87   ;ET OPERATION OR AVEC MEMOIRE ECRAN
0C88   ;ECRIRE DANS LA MEMOIRE ECRAN
0C89   ;TERMINE
```

0C8A SCR READ

Lors de l'appel, SCR READ s'attend à trouver dans le registre HL une adresse de la RAM vidéo.

```
0C8A   ;ALLER CHERCHER OCTET DE LA RAM VIDEO
0C8B   ;CALCULER CRAYON DE COULEUR
```

0C8E SCR INK ENCODE

Codage d'une Ink pour que tous les points d'image soient fixés sur cette Ink. Cette routine attend en entrée le crayon de couleur dans l'accumulateur et écrit en sortie le masque couleur dans l'accu.

```
0C8E   ;LIBERER REGISTRE BC
0C8F   ;LIBERER REGISTRE DE
0C90   ;CNV NUMERO PEN
0C93   ;NUMERO PEN DANS REGISTRE E
0C94   ;GET PIXEL MASK
0C97   ;COMPTEUR POUR MANIPULATION DE MASQUE
0C99   ;BIT 0 DU NUMERO PEN DANS LE CARRY
0C9B   ;ET DANS LE BIT 7 DE L'ACCU
0C9C   ;MANIPULER MASQUE BITS DE POINT
0C9E   ;SI BIT 0 = 1 ALLER CHERCHER BIT SUIVANT
0CA0   ;SINON RECONSTITUER MASQUE
0CA2   ;ALLER CHERCHER BIT SUIVANT
0CA4   ;ANCIEN CONTENU DE REGISTRE DE
0CA5   ;ANCIEN CONTENU DE REGISTRE BC
0CA6   ;C'EST TOUT
```

0CA7 SCR INK DECODE

Décodage d'une Ink. Cette routine attend le masque couleur dans l'accumulateur, en entrée, et écrit, en sortie, le crayon de couleur dans l'accu.

```
0CA7  ;LIBERER BC
0CA8  ;SAUVER LE CONTENU DE L'ACCUMULATEUR
0CA9  ;GET PIXEL MASK
0CAC  ;ALLER CHERCHER CONTENU DE L'ACCUMULATEUR
0CAD  ;CALC PEN NUMBER
0CB0  ;ALLER CHERCHER CONTENU DE BC
0CB1  ;TERMINE!
```

0CB2 CALC PEN-NUMBER

Le masque couleur est transmis à CALC PEN-NUMBER dans l'accumulateur et le masque bits dans le registre C. Lors de l'abandon de la routine, le code couleur figure dans le registre C.

```
0CB2  ;LIBERER REGISTRE DE
0CB3  ;INITIALISER LE COMPTEUR DE BITS
0CB6  ;ALLER CHERCHER BIT DANS LE MASQUE COULEUR
0CB7  ;ET TRANSFERER DANS D
0CB9  ;ALLER CHERCHER BIT DANS MASQUE COULEUR
0CBB  ;BIT = 0?
0CBD  ;TRANSFERER BIT DANS D
0CBF  ;COMPTEUR DE BIT - 1
0CC0  ;Y A-T-IL UN AUTRE BIT?
0CC2  ;ACCU DEVIENT NUMERO PEN
0CC3  ;CNV PEN-NUMBER
0CC6  ;RESTAURER ANCIEN CONTENU DE DE
0CC7  ;TERMINE!
```

0CC8 CNV PEN-NUMBER

```
0CC8  ;TRANSFERER ACCU DANS REGISTRE D
0CC9  ;SCR GET MODE
0CCC  ;RENVOYER VALEUR DANS L'ACCUMULATEUR
0CCD  ;RETOUR SI DIFFERENT MODE 0
```

0CCE ;DIVISER CONTENU ACCUMULATEUR
0CCF ;PAR QUATRE
0CD0 ;AJOUTER CARRY S'IL Y A LIEU
0CD2 ;DIVISER RESULTAT PAR DEUX
0CD3 ;ISOLER CARRY (CARRY = &FF)
0CD4 ;OPERATION AND AVEC SIX
0CD6 ;A = NUMERO PEN
0CD7 ;REGLE!

0CD8 REINITIALISATION DES COULEURS

0CD8 ;SOURCE: COULEURS DEFAUT
0CDB ;DESTINATION: MEMOIRE COULEUR SECONDES COULEURS
0CDE ;COMPTEUR: &22 OCTETS
0CE1 ;EXECUTER OPERATION DE COPIE
0CE3 ;ANNULER ACCU ET REINITIALISER LES FLAGS
0CE4 ;(FLAG JEU DE COULEURS ACTUEL)
0CE7 ;VALEUR DE DEPART POUR FLASH PERIODS

0CEA SCR SET FLASHING

Fixer les délais de clignotage pour la représentation des couleurs pour toutes les Inks et pour le cadre. La valeur pour la variable système FLASH PERIODS doit être transmise à SCR SET FLASHING dans le registre double HL.

0CEA ;(FLASH PERIODS)
0CED ;DEJA TERMINE

0CEE SCR GET FLASHING

0CEE ;(FLASH PERIODS)
0CF1 ;C'EST TOUT

0CF2 SCR SET INK

Affectation des deux couleurs utilisées pour représenter une Ink. Lors de l'appel de la routine, le numéro du crayon de couleur doit figurer dans l'accumulateur. SCR SET INK attend dans le registre B le code de la première couleur et dans le registre C le code de la seconde couleur.

```
0CF2   ;DECONNECTER LES BITS 4 A 7
0CF4   ;AUGMENTER L'ACCUMULATEUR DE 1
       ;LE NUMERO DE CRAYON DE COULEUR EST MAINTENANT
       ;COMPRIS ENTRE 1 ET 16
0CF5   ;SET COLOUR
```

0CF7 SCR SET BORDER

Affectation des deux couleurs utilisées pour représenter un cadre. SCR SET BORDER attend dans le registre B le code de la première couleur et dans le registre C le code de la seconde couleur.

```
0CF7   ;0 EST LE CRAYON DE COULEUR POUR BORDER
```

0CF8 ;SET COLOUR

Cette routine attend dans le registre B le code de la première couleur et dans le registre C le code de la seconde couleur.

```
0CF8   ;STOCKER LE NUMERO DU CRAYON DE COULEUR
0CF9   ;CODE PREMIERE COULEUR DANS ACCU
0CFA   ;ALLER CHERCHER ENTREE MATRICE COULEUR
0CFD   ;PREMIERE VALEUR COULEUR DANS B
0CFE   ;CODE SECONDE COULEUR DANS ACCU
0CFF   ;ALLER CHERCHER ENTREE MATRICE COULEUR
0D02   ;SECONDE VALEUR COULEUR DANS C
0D03   ;RENVOYER CODE DU CRAYON DE COULEUR DANS A
0D04   ;ALLER CHERCHER ADRESSE INK
0D07   ;PLACER SECONDE VALEUR COULEUR DANS LA RAM
0D08   ;ECHANGER POINTEUR
0D09   ;PLACER PREMIERE VALEUR COULEUR DANS LA RAM
0D0A   ;FLAG POUR NOUVELLES VALEURS COULEUR
```

```
0D0C  ;FLAG EST MIS
0D0F  ;REGLE
```

0D10 ALLER CHERCHER ENTREE MATRICE COULEUR

Cette routine détermine l'adresse d'une valeur de couleur à l'intérieur de la matrice couleur. Un code couleur est attendu dans l'accumulateur comme paramètre d'entrée. Lors de l'abandon de la routine, l'adresse de la valeur de couleur dans la matrice couleur figure dans le registre HL.

```
0D10  ;DANS L'INTERVALLE DE 0 A 31
0D12  ;AJOUTER OCTET FAIBLE DE L'ADRESSE DE BASE
0D14  ;ENVOYER RESULTAT DANS L
0D15  ;AJOUTER OCTET FORT DE L'ADRESSE DE BASE ET CARRY
0D17  ;RETRANCHER A NOUVEAU OCTET FAIBLE
0D18  ;OCTET FORT DE L'ADRESSE DANS H
0D19  ;C'EST TOUT
```

0D1A SCR GET INK

Aller chercher les deux couleurs utilisées pour représenter une Ink. En entrée, le numéro du crayon de couleur est attendu dans l'accumulateur. SCR GET INK fournit en sortie le code de la première couleur dans le registre B et le code de la seconde couleur dans le registre C.

```
0D1A  ;MASQUER LES BITS 4 A 7
0D1C  ;AUGMENTER ENSUITE DE UN
      ;LE NUMERO DE CRAYON DE COULEUR EST
      ;MAINTENANT COMPRIS ENTRE 1 ET 16
0D1D  ;GET COLOUR
```

0D1F SCR GET BORDER

Aller chercher les deux couleurs utilisées pour représenter un cadre. En sortie, SCR GET BORDER fournit le code de la première couleur dans le registre B et le code de la seconde couleur dans le registre C.

```
0D1F  ;0 EST LE CRAYON DE COULEUR POUR BORDER
```

0D20 GET COLOUR

GET COLOUR fournit en sortie le code de la première couleur dans le registre B et le code de la seconde couleur dans le registre C.

```
0D20   ;ALLER CHERCHER ADRESSE INK
0D23   ;PREMIERE VALEUR DE COULEUR DANS A
0D24   ;SECONDE VALEUR DE COULEUR DANS E
0D25   ;LOCALISER VALEUR DE COULEUR DANS LA TABLE
0D28   ;PREMIER CODE COULEUR DANS LE REGISTRE B
0D29   ;SECONDE VALEUR DE COULEUR DANS L'ACCU
0D2A   ;INITIALISER COUNTER
0D2C   ;ADRESSE DE LA MATRICE COULEUR DANS HL
0D2F   ;VALEUR DE COULEUR LOCALISEE?
0D30   ;ALORS TERMINE!
0D31   ;AUGMENTER POINTEUR DE TABLE ET
0D32   ;CODE COULEUR
0D33   ;POURSUIVRE LA RECHERCHE!
```

0D35 ALLER CHERCHER ADRESSE INK

ALLER CHERCHER ADRESSE INK attend le numéro du crayon de couleur dans l'accumulateur. Cette indication permet à la routine de déterminer où figurent dans la RAM les deux valeurs de couleur affectées au crayon de couleur. Lors de l'abandon de ALLER CHERCHER ADRESSE INK, DE contient l'adresse à laquelle figure la première valeur de couleur et HL contient l'adresse de la seconde valeur de couleur.

```
0D35   ;NUMERO DU CRAYON DE COULEUR
0D36   ;PRODUIRE L'OCTET FORT
0D38   ;MEMOIRE COULEUR PREMIERES COULEURS
0D3B   ;AJOUTER LE NUMERO DU CRAYON DE COULEUR
0D3C   ;ECHANGER RESULTAT DANS DE
0D3D   ;OPERANDE AUXILIAIRE POUR SECONDE VALEUR DE COULEUR
0D40   ;HL INDIQUE LA SECONDE VALEUR DE COULEUR
0D41   ;TERMINE
```

0D42 INSERT COLOUR EVENT BLOCK

0D42 ;EVENT BLOCK SET INKS
0D45 ;ADRESSE EVENT BLOCK SUR LA PILE
0D46 ;KL DEL FRAME FLY
0D49 ;FLASH INKS
0D4C ;SET INKS ON FRAME FLY
0D4F ;CHARGER &81 DANS LE REGISTRE B
0D51 ;RETIRER ADRESSE EVENT BLOCK DE LA PILE
0D52 ;KL NEW FRAME FLY

0D55 TERMINATE COLOUR EVENT BLOCK

0D55 ;EVENT BLOCK: SET INKS
0D58 ;KL DEL FRAME FLY
0D5B ;ALLER CHERCHER PARAMETRES DU JEU DE COULEURS ACTUEL
0D5E ;MC CLEAR INKS

0D61 SET INKS ON FRAME FLY

0D61 ;CURR. FLASH PERIOD
0D64 ;DECREMENTER FLASH PERIOD
0D65 ;FLASH INKS
0D67 ;POINTEUR SUR VALEUR DE COLONNE
0D68 ;ALLER CHERCHER FLAGS VALEUR DE COLONNE
0D69 ;FLAG VALEUR DE COLONNE POSITIF?
0D6A ;SI NON, ALORS TERMINE!
0D6B ;ALLER CHERCHER PARAMETRES DU JEU DE COULEURS ACTUEL
0D6E ;MC SET INKS
0D71 ;DELETE VALEUR DE COLONNE

0D73 FLASH INKS

0D73 ;ALLER CHERCHER PARAMETRES DU JEU DE COULEURS ACTUEL
0D76 ;(CURR FLASH PERIOD)
0D79 ;MC SET INKS
0D7C ;FLAG JEU DE COULEURS ACTUEL
0D7F ;JEU DE COULEURS ACTUEL DANS ACCU
0D80 ;FORMER COMPLEMENT A UN

```
0D81   ;RENVOYER RESULTAT DANS FLAG JEU DE COULEURS ACTUEL
0D82   ;ANNULER ACCU ET REINITIALISER FLAGS
0D83   ;ANNULER TOUS LES FLAGS VALEUR DE COLONNE
0D86   ;RETOUR
```

0D87 ALLER CHERCHER PARAMETRES DU JEU DE COULEURS ACTUEL

Cette routine fournit en sortie les Flash Periods dans l'accumulateur et l'adresse de la table de couleurs dans le registre DE.

```
0D87   ;MEMOIRE COULEURS PREMIERES COULEURS
0D8A   ;(FLAG JEU DE COULEURS ACTUEL)
0D8D   ;ACCU = 0?
0D8E   ;(FLASH PERIOD 1ERE COULEUR)
0D91   ;SI ACCU = 0, ALORS TERMINE!
0D92   ;MEMOIRE COULEURS SECONDE COULEUR
0D95   ;(FLASH PERIODS)
0D98   ;RETOUR
```

0D99 MATRICE COULEURS

```
0D99   14 04 15 1C 18 1D 0C 05
0DA1   0D 16 06 17 1E 00 1F 0E
0DA9   07 0F 12 02 13 1A 19 1B
0DB1   0A 03 0B 01 08 09 10 11
```

0DB9 SCR FILL BOX

Remplir la fenêtre indiquée avec une couleur déterminée (position exprimée en caractères, en fonction du mode). Le masque couleur est transmis à SCR FILL BOX dans l'accumulateur.

```
0DB9   ;MASQUE COULEURS DANS LE REGISTRE C
0DBA   ;COMPUTE WINDOW PARAMS
```

0DBD SCR FLOOD BOX

Remplir la fenêtre indiquée avec une couleur déterminée (les positions sont des adresses écran indépendantes du mode). Pour remplir la fenêtre, la routine doit recevoir dans le registre HL l'adresse écran à partir de laquelle doit se faire le remplissage. Le contenu du registre DE fixe la taille de la zone à remplir et le registre C contient le masque couleurs qui sera utilisé pour le remplissage.

```
0DBD  ;PLACER ADRESSE ECRAN SUR LA PILE
0DBE  ;NOMBRE D'OCTETS PAR LIGNE
0DBF  ;DETERMINER LA RETENUE
0DC2  ;PAS DE RETENUE
0DC4  ;NOMBRE D'OCTETS PAR LIGNE
0DC5  ;MASQUE COULEURS DANS LA MEMOIRE ECRAN
0DC6  ;SCR NEXT BYTE
0DC9  ;CONTINUER TANT QU'IL Y A ENCORE DES OCTETS
0DCB  ;LIGNE DE GRILLE SUIVANTE
0DCD  ;LIBERER LE REGISTRE BC
0DCE  ;LIBERER LE REGISTRE DE
0DCF  ;ECRIRE UN OCTET DANS LA MEMOIRE ECRAN
0DD0  ;SAUTER SI CETTE LIGNE
0DD1  ;A FINI D'ETRE TRAITEE
0DD3  ;OCTET FAIBLE COMPTEUR D'OCTETS A COPIER
0DD4  ;OCTET FORT COMPTEUR
0DD6  ;OCTET FORT ADRESSE ECRAN
0DD7  ;OCTET FAIBLE ADRESSE ECRAN
0DD8  ;ADRESSE OBJET POUR LDIR
0DD9  ;REMPLIR MEMOIRE ECRAN
0DDB  ;ANCIEN CONTENU DE REGISTRE DE
0DDC  ;ANCIEN CONTENU DE REGISTRE BC
0DDD  ;ALLER CHERCHER ADRESSE ECRAN SUR LA PILE
0DDE  ;SCR NEXT LINE
0DE1  ;S'IL Y A ENCORE DES LIGNES DE GRILLE,
0DE2  ;ALORS CONTINUER
0DE4  ;TRAVAIL ACCOMPLI
```

0DE5 SCR CHAR INVERT

Interchanger les couleurs de premier plan et de fond pour un caractère. La position du caractère est transmise à SCR CHAR INVERT dans le registre HL (H=colonne, L=ligne).

```
0DE5   ;COULEUR DU FOND DANS L'ACCU
0DE6   ;OPERATION XOR AVEC COULEUR DE PREMIER PLAN
0DE7   ;PLACER LE MASQUE D'INVERSION DANS C
0DE8   ;SCR CHAR POSITION
0DEB   ;IL Y A HUIT LIGNES DE GRILLE
0DED   ;ADRESSE ECRAN SUR LA PILE
0DEE   ;B=NOMBRE D'OCTETS PAR CARACTERE
       ;C=MASQUE D'INVERSION CALCULE
0DEF   ;LIRE OCTET DANS ADRESSE ECRAN
0DF0   ;INVERSER OCTET
0DF1   ;RENVOYER OCTET DANS ADRESSE ECRAN
0DF2   ;SCR NEXT BYTE
0DF5   ;CONTINUER TANT QU'IL Y A DES OCTETS
0DF7   ;B=NOMBRE D'OCTETS PAR CARACTERE
       ;C=MASQUE D'INVERSION CALCULE
0DF8   ;ALLER CHERCHER ADRESSE ECRAN SUR LA PILE
0DF9   ;SCR NEXT LINE
0DFC   ;LIGNE DE GRILLE=LIGNE DE GRILLE-1
0DFD   ;CONTINUER TANT QU'IL RESTE DES LIGNES DE GRILLE
0DFF   ;TERMINE
```

0E00 SCR HW ROLL

Décalage (machine) de l'écran d'une ligne vers le bas si B=0 et d'une ligne vers le haut si B<>0. L'accumulateur doit contenir la valeur définissant les couleurs que devra prendre la nouvelle ligne (vide).

```
0E00   ;COULEUR DE FOND DANS LE REGISTRE C
0E01   ;DIRECTION DE SCROLLING ET COULEUR SUR LA PILE
0E02   ;CHARGER VALEUR DEFAUT DANS DE
0E05   ;CHARGER VALEUR DEFAUT DANS REGISTRE B
0E07   ;FILL AREA
0E0A   ;RETIRER DIRECTION SCROLLING ET COULEUR DE LA PILE
0E0B   ;MC WAIT FLYBACK
```

0E0E	;DIRECTION DE SCROLLING DANS L'ACCUMULATEUR
0E0F	;MANIPULER FLAG ZERO
0E10	;DIRECTION DE DECALAGE HAUT
0E12	;DE = OFFSET
0E15	;ADD OFFSET
0E18	;INITIALISER DE
0E1B	;OCTETS PAR LIGNE
0E1D	;ADD OFFSET
0E1F	;DE = OFFSET
0E22	;ADD OFFSET
0E25	;DE = OFFSET
0E28	;OCTETS PAR LIGNE
0E2A	;POSITION SUR UNE LIGNE
0E2D	;AJOUTER OFFSET
0E2E	;A = OCTET FORT
0E2F	;PRENDRE EN COMPTE ADRESSE DE GRILLE
0E31	;ET ECRIRE EN RETOUR LE RESULTAT
0E32	;(OCTET FORT SCREEN START)
0E35	;OCTET FORT ADRESSE ECRAN
0E36	;CALCULER ET PLACER DANS REGISTRE H
0E37	;OCTETS PAR LIGNE
0E38	;IL Y A HUIT LIGNES DE GRILLE
0E3A	;SCR FLOOD BOX

0E3D ADD OFFSET

0E3D	;(POSITION SUR UNE LIGNE)
0E40	;AJOUTER ADRESSE DE BASE ET OFFSET
0E41	;SCR SET OFFSET

0E44 SCR SW ROLL

Décale une zone de l'écran par logiciel. A et B ont la même signification que pour SCR HW ROLL. H doit en plus contenir le numéro de colonne de la limite gauche de la zone à décaler et L la ligne limite supérieure, D la colonne droite et E la ligne inférieure de la zone.

Notez bien que les colonne et ligne 0 représentent le coin supérieur gauche de l'écran. Vous devez absolument contrôler par vous-même que les paramètres transmis marquent vraiment une zone à l'intérieur de la RAM vidéo.

```
0E44   ;SAUVER COULEUR DE FOND
0E45   ;A = DIRECTION DU SCROLLING
0E46   ;RESULTAT NUL?
0E47   ;SI 0 SCROLLING DESCENDANT
0E49   ;SAUVER X1,Y1
0E4A   ;COMPUTE WINDOW PARAMS
0E4D   ;ALLER CHERCHER X1 ET Y1
0E4E   ;Y1=Y1+1
0E4F   ;SCR CHAR POSITION
0E52   ;C = X2
0E53   ;A = Y2
0E54   ;Y2=Y2 MOINS LIGNES ECRAN (8
       ;LIGNES DE GRILLE)
0E56   ;RESULTAT (COUNTER) DANS B
0E57   ;RESULTAT NUL?
0E59   ;ALLER CHERCHER X2 ET Y2
0E5A   ;MC WAIT FLYBACK
0E5D   ;SAUVER X2,YCOUNTER
0E5E   ;SAUVER X1 ET Y1
0E5F   ;SAUVER X2 ET Y2
0E60   ;COPY RASTER LINE
0E63   ;ALLER CHERCHER X2 ET Y2
0E64   ;SCR NEXT LINE
0E67   ;DE = X2,Y2
0E68   ;ALLER CHERCHER X1 ET Y1
0E69   ;SCR NEXT LINE
0E6C   ;ALLER CHERCHER X2 ET MODY2 (COUNTER)
0E6D   ;COUNTER=0? (LIGNES DE GRILLE)
0E6F   ;RETENIR X2,Y2
0E70   ;DANS HL
0E71   ;OCTET PAR LIGNE DE GRILLE
0E72   ;E = UNE LIGNE (8 LIGNES DE GRILLE)
0E74   ;ALLER CHERCHER COULEUR DU PAPIER
0E75   ;C = COULEUR DU PAPIER
0E76   ;SCR FLOOD BOX
0E79   ;SAUVER X1 ET Y1
0E7A   ;SAUVER X2 ET Y2
```

```
0E7B   ;CALC WIN PARAMS
0E7E   ;NOMBRE D'OCTETS DANS LA FENETRE (DIRECTION X)
0E7F   ;A = LIGNES (DIRECTION Y)
0E80   ;MOINS 8 LIGNES DE GRILLE
0E82   ;RESULTAT DANS B
0E83   ;ALLER CHERCHER X2 ET Y2
0E84   ;ECHANGER AVEC X1,Y1
0E85   ;COPIER LIGNE DE GRILLE
0E87   ;SAUVER NOMBRE DE LIGNES (COUNTER)
0E88   ;L = X1
0E89   ;D = Y1
0E8A   ;X1 = X1 MOINS 1
0E8B   ;SCR CHAR POSITION
0E8E   ;DE = X1,Y1
0E8F   ;SCR CHAR POSITION
0E92   ;ALLER CHERCHER COUNTER
0E93   ;MC WAIT FLYBACK
0E96   ;SCR PREV LINE
0E99   ;SAUVER X2,Y2
0E9A   ;DE = X2,Y2
0E9B   ;SCR PREV LINE
0E9E   ;SAUVER X1,Y1
0E9F   ;SAUVER COUNTER
0EA0   ;COPY RASTER LINE
0EA3   ;ALLER CHERCHER COUNTER
0EA4   ;ALLER CHERCHER X1,Y1
0EA5   ;ALLER CHERCHER X2,Y2
0EA6   ;TERMINE?
0EA8   ;KILL UPPER LINE (8 LIGNES DE GRILLE)
```

0EAA COPY RASTER LINE

COPY RASTER LINE copie une ligne de grille complète dans une autre zone. HL doit contenir ici l'adresse de la ligne source et DE l'adresse de la ligne de destination. Le registre C fixe le nombre d'octets à copier.

```
0EAA   ;COUNTER INIT
0EAC   ;APPELER TEST ADRESSE DE GRILLE
0EAF   ;APPARU
0EB1   ;APPELER TEST ADRESSE DE GRILLE
```

```
0EB4   ;INDIRECT COPY?
0EB6   ;SAUVER LE NOMBRE D'OCTETS
0EB7   ;CALCULER LONGUEUR
0EB8   ;VOIR 0EB7
0EB9   ;LONGUEUR DANS C
0EBA   ;COPIER LA PREMIERE PARTIE DE LA LIGNE
0EBC   ;ALLER CHERCHER NOMBRE D'OCTETS
0EBD   ;CALCULER NOMBRE D'OCTET A COPIER
0EBE   ;VOIR 0EBD
0EBF   ;VOIR 0EBD
0EC0   ;VOIR 0EBD
0EC1   ;RESULTAT DANS ACCU
0EC2   ;RECONSTITUER LES BITS DE L'ADRESSE DE GRILLE
0EC4   ;RESULTAT DANS H
0EC5   ;COPIER SECONDE PARTIE DE LA LIGNE
0EC7   ;APPELER TEST ADRESSE DE GRILLE
0ECA   ;COPY STRAIGHT LINE
0ECC   ;SAUVER NOMBRE D'OCTETS
0ECD   ;CALCULER LONGUEUR
0ECE   ;VOIR 0ECE
0ECF   ;LONGUEUR DANS C
0ED0   ;COPIER SECONDE PARTIE DE LA LIGNE
0ED2   ;ALLER CHERCHER NOMBRE D'OCTETS
0ED3   ;CALCULER NOMBRE D'OCTETS COPIES
0ED4   ;VOIR 0ED3
0ED5   ;VOIR 0ED3
0ED6   ;VOIR 0ED3
0ED7   ;RESULTAT DANS ACCU
0ED8   ;RECONSTITUER BITS D'ADRESSE DE GRILLE
0EDA   ;RESULTAT DANS D
0EDB   ;COPIE NORMALE
0EDD   ;TERMINE
```

0EDE COPY STRAIGHT LINE

COPY STRAIGHT LINE copie une ligne de grille complète dans une autre zone. HL doit contenir ici l'adresse de la ligne source et DE l'adresse de la ligne de destination. Le registre C fixe le nombre d'octets à copier.

```
0EDE  ;ALLER CHERCHER LONGUEUR DE LA LIGNE
0EDF  ;ALLER CHERCHER OCTET SOURCE
0EE0  ;LE COPIE DANS OCTET DE DESTINATION
0EE1  ;SCR NEXT BYTE
0EE4  ;HL = POINTEUR DESTINATION
0EE5  ;SCR NEXT BYTE
0EE8  ;HL = POINTEUR SOURCE
0EE9  ;LIGNE COPIEE?
0EEB  ;TERMINE
```

0EEC RA-TEST

RA-TEST détermine si une retenue s'est produite dans les Raster Adress Bits (bits d'adresse de grille). La routine attend comme paramètres en entrée une adresse écran dans HL et dans DE ainsi que la longueur de ligne dans C. Si une retenue a eu lieu, le bit Carry se trouve mis (=1) après abandon de RA-TEST.

```
0EEC  ;A = NOMBRE D'OCTETS
0EED  ;DE = 1ERE ADRESSE VIDEO
0EEE  ;COUNTER -1
0EEF  ;AJOUTER OCTET FAIBLE DE LA SECONDE ADRESSE VIDEO AU
      ;COUNTER
0EF0  ;TERMINE?
0EF1  ;ALLER CHERCHER OCTET FORT DE LA SECONDE ADRESSE ECRAN
0EF2  ;RETENUE ?
0EF4  ;VOIR 0EF2
0EF6  ;TERMINE?
0EF7  ;CARRY = MIS (OUT PARAMS)
0EF8  ;TERMINE!
```

0EF9 SCR UNPACK

SCR UNPACK agrandit la matrice de caractère pour MODE 0 et 1. La routine attend l'adresse de la matrice comprimée dans le registre HL et dans le registre DE l'adresse de destination pour la matrice calculée.

```
0EF9  ;SCR GET MODE
0EFC  ;MODE 0 FIXE
```

```
0EFE   ;MODE 1 FIXE
0F00   ;COMPTEUR: HUIT OCTETS
0F03   ;EXECUTER OPERATION DE COPIE
0F05   ;C'EST TOUT

0F06   ;COMPTEUR DE BITS ET MASQUE DANS BC (MODE 0)
0F09   ;JUMP START
0F0B   ;COMPTEUR DE BITS ET MASQUE DANS BC (MODE 1)
0F0E   ;INITIALISER COMPTEUR (COPIER ;8 OCTETS)
0F10   ;SAUVER COMPTEUR
0F11   ;SAUVER POINTEUR POUR MATRICE COMPACTE
0F12   ;L = ELEMENT DE MATRICE COMPRIME
0F13   ;RANGER COMPTEUR DE BITS
0F14   ;ANNULER ACCU
0F15   ;ALLER CHERCHER BIT DANS MATRICE COMPRIMEE
0F17   ;Y A-T-IL UN BIT?
0F19   ;FUSIONNER MATRICE
0F1A   ;ALLER CHERCHER BIT DANS MATRICE
0F1C   ;BIT PRESENT?
0F1E   ;MEMOIRE OCTET NON COMPRIME
0F1F   ;POINTEUR SUR OCTET SUIVANT
0F20   ;TOUS LES BITS ONT ETE TRAITES?
0F22   ;ALLER CHERCHER COMPTEUR DE BITS
0F23   ;ALLER CHERCHER POINTEUR POUR MATRICE COMPRIMEE
0F24   ;INDIQUER LIGNE SUIVANTE DE LA MATRICE
0F25   ;ALLER CHERCHER COUNTER
0F26   ;COUNTER-1
0F27   ;MATRICE COPIEE ET CONVERTIE?
0F29   ;TERMINE!
```

0F2A SCR REPACK

Ramener la matrice de caractère dans sa forme d'origine. La routine attend la colonne de la matrice dans le registre H et la ligne de la matrice dans le registre H. L'accumulateur doit en outre contenir le numéro PEN et le registre DE l'adresse de destination de la matrice de caractère comprimée.

```
0F2A   ;COULEUR DE FOND DANS LE REGISTRE C
0F2B   ;SCR CHAR POSITION
```

```
0F2E   ;SCR GET MODE
0F31   ;IL Y A HUIT LIGNES DE GRILLE
0F33   ;MODE 0 FIXE
0F35   ;MODE 1 FIXE
0F37   ;REMPLIR ACCU AVEC LA MEMOIRE ECRAN
0F38   ;OPERATION XOR AVEC COULEUR DE FOND
0F39   ;FORMER COMPLEMENT A UN
0F3A   ;ECRIRE RESULTAT DANS (DE)
0F3B   ;AUGMENTER POINTEUR
0F3C   ;SCR NEXT LINE
0F3F   ;TANT QU'IL Y AURA ENCORE DES LIGNES DE GRILLE
0F41   ;TERMINE

0F42   ;SAUVER LE CONTENU DE TOUS LES REGISTRES
0F43   ;VOIR 0F42
0F44   ;VOIR 0F42
0F45   ;CONVERTIR PREMIER OCTET
0F48   ;SCR NEXT BYTE
0F4B   ;CONVERTIR SECOND OCTET
0F4E   ;ACCU = ELEMENT DE MATRICE
0F4F   ;ALLER CHERCHER ADRESSE DE DESTINATION
0F50   ;ECRIRE ELEMENT DE MATRICE DANS ADRESSE DE DESTINATION
0F51   ;INDIQUER ADRESSE SUIVANTE
0F52   ;ALLER CHERCHER COLONNE ET LIGNE
0F53   ;SCR NEXT LINE
0F56   ;ALLER CHERCHER COMPTEUR
0F57   ;TOUS LES OCTETS ONT ETE TRANSFERES?
0F59   ;SI OUI ALORS TERMINE

0F5A   ;C = MASQUE (MODE 0)
0F5C   ;B = COMPTEUR DE BITS
0F5E   ;ALLER CHERCHER OCTET DANS LA RAM VIDEO
0F5F   ;MASQUE COULEUR DE FOND
0F60   ;FUSION AVEC MASQUE
0F61   ;POINT = COULEUR PEN?
0F63   ;SI NON ALORS ACTIVER POINT
0F64   ;INTEGRER POINT DANS MATRICE
0F66   ;VOIR 0F64
0F68   ;TOUS LES BITS ONT ETE TRANSFERES
0F6A   ;RETOUR

0F6B   ;SAUVER LE CONTENU DE TOUS LES REGISTRES
```

0F6C ;VOIR 0F6B
0F6D ;VOIR 0F6B
0F6E ;INITIALISER COMPTEUR
0F70 ;ALLER CHERCHER OCTET DANS RAM VIDEO
0F71 ;MASQUER COULEUR DE FOND
0F72 ;FUSION AVEC MASQUE
0F74 ;POINT = COULEUR PEN?
0F76 ;SI NON, ALORS ACTIVER POINT
0F77 ;INTEGRER POINT DANS MATRICE
0F79 ;VOIR 0F77
0F7A ;VOIR 0F77
0F7B ;VOIR 0F77
0F7D ;Y A-T-IL UN POINT?
0F7F ;SI NON, ACTIVER POINT
0F80 ;ET INTEGRER DANS LA MATRICE
0F82 ;SCR NEXT BYTE
0F85 ;TOUS LES BITS ONT ETE TRANSFERES?
0F87 ;A = ELEMENT DE MATRICE
0F88 ;ALLER CHERCHER POINTEUR DE MATRICE
0F89 ;ECRIRE OCTET DANS MATRICE
0F8A ;POINTEUR SUR OCTET SUIVANT DE LA MATRICE
0F8B ;ALLER CHERCHER ADRESSE VIDEO
0F8C ;SCR NEXT LINE
0F8F ;ALLER CHERCHER COMPTEUR
0F90 ;TOUTES LES LIGNES DE GRILLE ONT ETE TRANSFEREES?
0F92 ;TERMINE!

0F93 SCR HORIZONTAL

Tracer une ligne horizontale. La routine attend X1 dans le registre DE, X2 dans le registre BC et la coordonnée Y dans le registre HL. L'accumulateur doit contenir le masque couleurs.

0F93 ;MASK HANDLING
0F96 ;DRAW H-LINE
0F99 ;A SUIVRE

0F9B SCR VERTICAL

Tracer une ligne verticale. La routine attend Y1 dans le registre HL, Y2 dans le registre BC et la coordonnée X dans le registre DE. L'accumulateur doit contenir le masque couleurs.

```
0F9B   ;MASK HANDLING
0F9E   ;DRAW V-LINE
0FA1   ;ANCIEN MASQUE DANS LE REGISTRE HL
0FA4   ;OCTET FAIBLE ANCIEN MASQUE DANS ACCU
0FA5   ;(GRA PEN)
0FA8   ;OCTET FORT ANCIEN MASQUE DANS ACCU
0FA9   ;MASK-PARAM
0FAC   ;TERMINE!
```

0FAD MASK HANDLING

```
0FAD   ;LIBERER REGISTRE HL
0FAE   ;(GRA PEN)
0FB1   ;(GRA PEN)
0FB4   ;PARAM. MASQUE DANS ACCU
0FB7   ;PARAM. MASQUE DANS H
0FB8   ;FIXER TOUS LES BITS
0FBA   ;DONNE UNE LIGNE CONTINUE
0FBD   ;CONSERVER ANCIEN MASQUE
0FC0   ;ANCIEN CONTENU DE REGISTRE HL
0FC1   ;C'EST TOUT
```

0FC2 DRAW H-LINE

```
0FC2   ;METTRE LE CARRY POUR H-LINE
0FC3   ;ALLER CHERCHER PARAMETRES
0FC6   ;CONSTITUER MASQUE DE LIGNE
0FC8   ;ALLER CHERCHER MASQUE POINTS
0FC9   ;BIT = PEN-COL
0FCB   ;NOMBRE DE POINTS-1
0FCC   ;TOUS LES POINTS ONT ETE TRANSFERES?
0FCE   ;NOMBRE DE POINTS+1
0FCF   ;TOUS LES POINTS ONT ETE TRANSFERES?
0FD1   ;CONSTITUER MASQUE DE POINTS
0FD3   ;LIMITE ATTEINTE?
```

```
0FD5   ;PIXEL HIGH?
0FD7   ;PENCOL OK?
0FD9   ;SI NON FIXER PENCOL
0FDA   ;CONSTITUER MASQUE POINTS
0FDC   ;ALLER CHERCHER POINT SUIVANT
0FDE   ;NOMBRE DE POINTS-1
0FDF   ;TOUS LES POINTS ONT ETE TRANSFERES?
0FE1   ;NOMBRE DE POINTS+1
0FE2   ;PENCOL OK?
0FE4   ;CONSTITUER MASQUE POINTS
0FE6   ;PENCOL OK?
0FE8   ;PIXEL HIGH?
0FEA   ;PENCOL OK?
0FEC   ;SI NON FIXER PENCOL
0FED   ;CONSTITUER MASQUE POINTS
0FEF   ;BIT = PEN-COL
0FF1   ;SAUVER MASQUES
0FF2   ;ALLER CHERCHER MASQUE POINTS
0FF3   ;(GRA PAPER)
0FF6   ;RANGER DANS B
0FF7   ;ALLER CHERCHER MODE DE FOND
0FFA   ;LOCALISER MODE
0FFB   ;TRANSFERER OCTET PRET DANS LA RAM VIDEO
0FFD   ;SAUVER MASQUES
0FFE   ;ALLER CHERCHER MASQUE POINTS
0FFF   ;(GRA PAPER)
1002   ;TRANSFERER DANS B
1003   ;FLAG-INIT
1007   ;ALLER CHERCHER MASQUES
1008   ;LIMITE DEPASSEE?
100A   ;SCR NEXT BYTE
100D   ;TOUS LES POINTS ONT ETE TRANSFERES
100E   ;VOIR 100D
100F   ;VOIR 100D
1011   ;A=LINE-MASK
1012   ;RANGER MASQUE LIGNE
1015   ;TERMINE!
```

1016 DRAW V-LINE

```
1016   ;FIXER CARRY POUR H-LINE
1017   ;ALLER CHERCHER PARAMETRES
```

```
101A    ;CONSTITUER MASQUE DE LIGNE
101C    ;(GRA PEN)
101F    ;BIT = PEN-COL
1021    ;MODE FOND?
1024    ;MODE TRANSPARENT ACTIVE?
1025    ;SI NON CHANGER MASQUE
1027    ;(GRA PAPER)
102A    ;SAUVER MASQUES
102B    ;ALLER CHERCHER MASQUE COLONNE
102C    ;SCR WRITE
102F    ;ALLER CHERCHER MASQUES
1030    ;SCR PREV LINE
1033    ;NEXT PIXEL
1034    ;VOIR 1033
1036    ;TOUS LES POINTS ONT ETE TRANSFERES?
1037    ;VOIR 1036
1039    ;VOIR 1036
103B    ;SAUVER COORDONNEE Y
103C    ;V-LINE?
103E    ;TRANSFERER OCTET FORT DE LA COORDONNEE X DANS HL
103F    ;TRANSFERER OCTET FAIBLE DE LA COORDONNE X DANS HL
1040    ;ELIMINER CARRY POUR SBC
1041    ;DETERMINER COORDONNEE
1043    ;VOIR 1041
1046    ;SYNCHRONISER LONGUEUR
1047    ;VOIR 1046
1048    ;ET RANGER
1049    ;SCR DOT POSITION
104C    ;ALLER CHERCHER MASQUE LIGNE
104F    ;TRANSFERER DANS REGISTRE B
1050    ;ALLER CHERCHER LONGUEUR SYNCHRONISEE
1051    ;C'EST TOUT
```

1052 COULEURS DEFAUT

```
1052    04 04 0A 13 0C 0B 14 15
105A    0D 06 1E 1F 07 12 19 04
1062    17 04 04 0A 13 0C 0B 14
106A    15 0D 06 1E 1F 07 12 19
1072    0A 07
```

9.6.3 Text Screen (TXT)

Comme son nom l'indique, ce pack est chargé de la gestion des textes. Cela comprend l'organisation des huit fenêtres, la représentation de caractères sur l'écran ainsi que la gestion du curseur. Lors de l'exécution de ces tâches, TEXT SCREEN utilise à fond les possibilités du SCREEN PACK. Le SCREEN PACK est donc la force de travail de TEXT SCREEN.

Immédiatement après avoir été appelé, TEXT SCREEN commence par l'initialisation complète du pack. Celle-ci consiste notamment à copier les indirections du TEXT SCREEN dans la mémoire RAM. Ces indirections sont :

```
BDCD        ;TXT DRAW/UNDRAW CURSOR
            ;Fixer/annuler le curseur.

BDD0        ;TXT DRAW/UNDRAW CURSOR
            ;Fixer/annuler le curseur.

BDD3        ;TXT WRITE CHAR
            ;Ecrire un caractère sur l'écran

BDD6        ;TXT UNWRITE CHAR
            ;lire un caractère de l'écran.

BDD9        ;TXT OUT ACTION
            ;Sortie d'un caractère sur l'écran
            ;ou exécution d'un code de commande
```

La table de saut des caractères de commande est la seconde table copiée dans la RAM. Cette table sera placée à partir de l'adresse &B763.

A chacune des huit fenêtres correspond un bloc de paramètres d'une taille de &0E octets. Ce bloc est également placé sur sa valeur défaut par TEXT SCREEN. Les paramètres de fenêtre sont tout d'abord placés sur leur "valeur de départ" par la routine FIXER PARAMETRES DEFAUT. Les informations nécessaires pour la sélection des couleurs sont transmises à FIXER PARAMETRES DEFAUT dans le registre double HL. H contient la valeur pour la couleur de fond et L la valeur pour la couleur de premier plan.

Une fois qu'un bloc de paramètres a été initialisé de cette façon, ce bloc est rapporté aux huit fenêtres avec RESET PARAMS (TOUTES LES FENETRES). L'initialisation de TEXT SCREEN est alors achevée.

Les coordonnées exigées ou fournies dans les routines curseur doivent être comprises comme des indications logiques, c'est-à-dire qu'elles se rapportent à la fenêtre actuelle. Les coordonnées 1,1 représentent ici le coin supérieur de la fenêtre. Si vous voulez, par exemple avec TXT SET CURSOR, positionner le curseur en dehors de la fenêtre, il sera automatiquement fixé sur la position la plus proche possible à l'intérieur de la fenêtre si le curseur est activé ou s'il s'agit ensuite de représenter un caractère. La position actuelle (que vous pouvez obtenir avec TXT GET CURSOR) est alors également modifiée de ce fait.

Si le curseur est désactivé, la nouvelle position souhaitée est acceptée dans un premier temps jusqu'à ce qu'un caractère soit représenté ou bien jusqu'à ce que le curseur soit activé.

1074 TXT INITIALISE

Initialisation complète du TEXT PACK.

```
1074    ;TXT RESET
1077    ;ACCUMULATEUR ANNULE
1078    ;EFFACER LE FLAG MATRICE UTILISATEUR
107B    ;PAPER=0 ET PEN=1
107E    ;FIXER LES PARAMETRES DEFAUT
1081    ;RESET PARAMS (TOUTES LES FENETRES)
```

1084 TXT RESET

Les indirections du TEXT PACK sont copiées dans la RAM (à partir de l'adresse &BDCD).

```
1084    ;RESTORE TXT INDIRECTIONS
1087    ;MOVE (HL+3) DANS ((HL+1)), CNT=(HL)
108A    ;COPIER SAUTS CARACTERES DE COMMANDE DEFAUT
108D    ;15 OCTETS
```

108E ;ADRESSE DE DESTINATION

1090 ;CODE POUR JP
1091 ;TXT DRAW/UNDRAW CURSOR

1093 ;CODE POUR JP
1094 ;TXT DRAW/UNDRAW CURSOR

1096 ;CODE POUR JP
1097 ;TXT WRITE CHAR

1099 ;CODE POUR JP
109A ;TXT UNWRITE CHAR

109C ;CODE POUR JP
109D ;TXT OUT ACTION

109F RESET PARAMS (TOUTES LES FENETRES)

FIXER LES PARAMETRES DEFAUT (&1139) fixe les paramètres de fenêtre sur leur valeur initiale. Ceci fait, ces paramètres sont rapportés aux huit fenêtres (voir &107E et &1081).

109F ;NOMBRE DE FENETRES
10A1 ;DESTINATION: DEBUT DES PARAMETRES FENETRE 0
10A4 ;SOURCE: DIVERS PARAMETRES POUR FENETRE
;&B726 CURSOR POSITION (ROW, COL)
;&B728 SCROLL FLAG
;&B729 FENETRE HAUT
;&B72A FENETRE GAUCHE
;&B72B FENETRE BAS
;&B72C FENETRE DROITE
;&B72D ROLL COUNT
;&B72E FLAG CURSEUR
;&B72F PEN
;&B730 PAPER
;&B731 BACKGROUND MODE
;&B733 GRAPH CHAR WRITE MODE
;&B734 1ER CARACTERE MATRICE UTILISATEUR
10A7 ;COMPTEUR: NOMBRE D'OCTETS A COPIER
10AA ;EXECUTER OPERATION DE COPIE

10AC ;FENETRE SUIVANTE
10AD ;SAUTER SI ENCORE UNE FENETRE
10AF ;(FENETRE ECRAN ACTUELLE)
10B2 ;TERMINE

10B3 DECODAGE DES COULEURS

Lors de l'abandon de cette routine, le registre contient le code de la fenêtre écran actuelle.

10B3 ;(FENETRE ECRAN ACTUELLE)
10B6 ;ACCU DISPONIBLE POUR REUTILISATION ULTERIEURE
10B7 ;IL Y A HUIT FENETRES
10B9 ;ACCU FAIT OFFICE DE COMPTEUR
10BA ;COMPTEUR-1 (FENETRES 0 A 7)
10BB ;TXT STR SELECT
10BE ;TXT DRAW/UNDRAW CURSOR
10C1 ;TXT GET PAPER
10C4 ;PAPER ACTUEL
10C7 ;TXT GET PEN
10CA :(PEN ACTUEL)
10CD ;SAUTER SI ENCORE UNE FENETRE
10CF ;FENETRE ECRAN ACTUELLE DANS ACCU
10D0 ;TERMINE

10D1 PARAMETRES FENETRE SUR DEFAUT

Le numéro de fenêtre est transmis à la routine dans l'accumulateur.

10D1 ;NUMERO DE FENETRE DANS C
10D2 ;IL Y A HUIT FENETRES
10D4 ;ACCU FAIT OFFICE DE COMPTEUR
10D5 ;COMPTEUR-1 (FENETRES 0 A 7)
10D6 ;TXT STR SELECT
10D9 ;SAUVER LE CONTENU DES REGISTRES
10DA ;(TXT PEN ACTUEL)
10DD ;FIXER LES PARAMETRES DEFAUT
10E0 ;RESTAURER LE CONTENU DES REGISTRES
10E1 ;SAUTER SI ENCORE UNE FENETRE
10E3 ;NUMERO DE FENETRE DANS ACCU

10E4 TXT STR SELECT

TXT STR SELECT permet de sélectionner une fenêtre de texte. Le numéro de la nouvelle fenêtre de texte doit figurer dans l'accumulateur lors de l'appel de TXT STR SELECT. Lors de l'abandon de la routine, le numéro de la fenêtre de texte qui était activée avant l'appel figure dans l'accumulateur.

```
10E4   ;NUMERO DE FENETRE DANS L'INTERVALLE DE 0 A 7
10E6   ;FENETRE ECRAN ACTUELLE
10E9   ;CETTE FENETRE EST-ELLE DEJA ACTIVEE?
10EA   ;ALORS IL N'Y A PLUS RIEN A FAIRE
10EB   ;SAUVER LE CONTENU DES REGISTRES
10EC   ;SAUVER LE CONTENU DES REGISTRES
10ED   ;NUMERO DE LA FENETRE ECRAN ACTUELLE
       ;DANS HL
10EE   ;NUMERO DE LA NOUVELLE FENETRE
10EF   ;NUMERO EGALEMENT DANS B
10F0   ;ANCIEN NUMERO DANS L'ACCUMULATEUR
10F1   ;ADR. PARAMETRES FENETRE DANS DE
10F4   ;LDIR CNT=14
10F7   ;NOUVEAU NUMERO DANS L'ACCUMULATEUR
10F8   ;ADR. PARAMETRES FENETRE DANS DE
10FB   ;ECHANGER POUR L'OPERATION DE COPIE
10FC   ;LDIR CNT=14
10FF   ;L'ANCIEN NUMERO DE FENETRE EST
       ;FOURNI DANS L'ACCUMULATEUR
1100   ;RESTAURER LE CONTENU DES REGISTRES
1101   ;RESTAURER LE CONTENU DES REGISTRES
1102   ;RETOUR
```

1103 TXT SWAP STREAMS

Les paramètres (couleurs, limites de fenêtre, etc.) de deux fenêtres sont interchangés. TXT SWAP STREAMS a besoin pour cela des numéros des deux fenêtres dont les paramètres doivent être interchangés. Le numéro d'une fenêtre figure, lors de l'appel de la routine dans le registre B, le numéro de l'autre dans le registre C.

```
1103  ;(FENETRE ECRAN ACTUELLE)
1106  ;SAUVER SUR LA PILE
1107  ;2ND NUMERO DE FENETRE DANS ACCU
1108  ;TXT STR SELECT
110B  ;1ER NUMERO DE FENETRE DANS ACCU
110C  ;(FENETRE ECRAN ACTUELLE)
110F  ;ADR. PARAMETRES FENETRE DANS DE
1112  ;SAUVER ADR. PARAMETRES FENETRE
1113  ;2ND NUMERO DE FENETRE DANS ACCU
1114  ;ADR. PARAMETRES FENETRE DANS DE
1117  ;ALLER CHERCHER ADR. PARAMETRES FENETRE
1118  ;LDIR CNT=14
111B  ;ALLER CHERCHER FENETRE ECRAN ACTUELLE
111C  ;TXT STR SELECT
```

111E LDIR CNT=14

Copie 14 octets (paramètres pour fenêtre).

Le nombre d'octets à copier est fixé sur 14 (BC), la source et la destination peuvent, par contre, être choisies librement. Lors de l'appel de la routine, l'adresse source doit être chargée dans HL et DE doit contenir l'adresse de destination. Ce n'est que de cette façon que LDIR pourra être exécuté correctement. Le contenu de BC avant appel de la routine est restauré lors de l'abandon de celle-ci.

```
111E  ;SAUVER BC POUR REUTILISATION ULTERIEURE
111F  ;COMPTEUR: NOMBRE D'OCTETS A COPIER
1122  ;EXECUTER L'OPERATION DE COPIE
1124  ;BC COMME AVANT L'ENTREE
1125  ;RETOUR
```

1126 ADR. PARAMETRES FENETRE DANS DE

Si un numéro de fenêtre est transmis à cette routine dans l'accumulateur, la routine fournira lors de l'abandon l'adresse des paramètres correspondants dans le registre DE. Le registre HL contiendra l'adresse des paramètres correspondant à la fenêtre actuelle.

1126 ;NUMERO DE FENETRE ENTRE 0 ET 7
1128 ;SAUVER A DANS E
1129 ;2A (RESULTAT EN A)
112A ;3A (RESULTAT EN A)
112B ;6A (RESULTAT EN A)
112C ;7A (RESULTAT EN A)
112D ;14A (RESULTAT EN A)
112E ;AJOUTER OCTET FAIBLE DE L'ADRESSE DE BASE
1130 ;OCTET FAIBLE DE LA TABLE DE PARAMETRES
1131 ;EVENTUEL DEPASSEMENT D'OCTET FAIBLE
1133 ;PRENDRE EN COMPTE POUR CALCULER L'OCTET FORT
;DE LA TABLE DE PARAMETRES
1134 ;DE=ADRESSE INITIALE DE LA TABLE DE PARAMETRES
;(&0E OCTETS)
1135 ;POSITION CURSEUR ACTUELLE (ROW, COL)
1138 ;TERMINE

1139 FIXER PARAMETRES DEFAUT

Cette routine place les paramètres de fenêtre sur leur valeur "initiale". Les informations nécessaires pour la sélection de couleurs sont transmises à FIXER PARAMETRES DEFAUT dans le registre double HL. H contient la valeur pour la couleur de fond et L la valeur pour la couleur de premier plan.

1139 ;PAPER DANS D ET PEN DANS E
113A ;VALEUR POUR FLAG CURSEUR ACTUEL
113C ;(TXT FLAG CURSEUR ACTUEL)
113F ;PAPER DANS ACCU
1140 ;TXT SET PAPER
1143 ;PEN DANS ACCU
1144 ;TXT SET PEN
1147 ;EFFACER ACCU ET REINITIALISER FLAGS
1148 ;TXT SET GRAPHIC
114B ;TXT SET BACK
114E ;LIMITE SUPERIEURE GAUCHE DE FENETRE
1151 ;LIMITE INFERIEURE DROITE DE FENETRE
1154 ;TXT WIN ENABLE
1157 ;TXT VDU ENABLE

115A TXT SET COLUMN

TXT SET COLUMN fixe la position horizontale du curseur. La valeur de colonne doit être transmise dans l'accumulateur.

```
115A    ;DIMINUER DE UN
115B    ;FENETRE ACTUELLE GAUCHE
115E    ;DONNE LA POSITION ABSOLUE
115F    ;(POSITION CURSEUR ACTUELLE (ROW, COL))
1162    ;NOUVELLE POSITION DANS LA COLONNE
1163    ;PASSER AU POSITIONNEMENT DU CURSEUR
```

1165 TXT SET ROW

TXT SET ROW fixe la position verticale du curseur. L'accumulateur doit contenir la valeur de ligne.

```
1165    ;DIMINUER DE UN
1166    ;FENETRE ACTUELLE HAUT
1169    ;DONNE LA POSITION ABSOLUE
116A    ;(POSITION CURSEUR ACTUELLE (ROW, COL))
116D    ;NOUVELLE POSITION SUR LA LIGNE
116E    ;PASSER AU POSITIONNEMENT DU CURSEUR
```

1170 TXT SET CURSOR

TXT SET CURSOR positionne le curseur. Avant appel de la routine, H doit recevoir la valeur de colonne et L la valeur de ligne.

```
1170    ;FENETRE ACTUELLE HAUT, GAUCHE + HL
1173    ;TXT DRAW/UNDRAW CURSOR
1176    ;(POSITION CURSEUR ACTUELLE (ROW, COL))
1179    ;TXT DRAW/UNDRAW CURSOR
```

117C TXT GET CURSOR

Cette routine permet de déterminer la position actuelle du curseur. Lors de l'abandon de TXT GET CURSOR, H contient la valeur de colonne et L la valeur de ligne.

L'accumulateur contient en outre le ROLL COUNT (compteur de scrolling) actuel.

```
117C   ;(POSITION CURSEUR ACTUELLE (ROW, COL))
117F   ;FENETRE ACTUELLE HAUT, GAUCHE - HL
1182   ;(ROLL COUNT ACTUEL)
1185   ;TERMINE
```

1186 FENETRE ACTUELLE HAUT, GAUCHE + HL

Cette routine convertit une position relative en une position absolue. Lors de l'appel comme après l'abandon de la routine, H contient l'indication de la colonne et L l'indication de la ligne.

```
1186   ;(FENETRE ACTUELLE HAUT)
1189   ;FENETRE ACTUELLE HAUT - 1 DONNE OFFSET
118A   ;OFFSET + LIGNE
118B   ;DONNE POSITION DE LIGNE ABSOLUE
118C   ;(FENETRE ACTUELLE GAUCHE)
118F   ;FENETRE ACTUELLE GAUCHE - 1 DONNE OFFSET
1190   ;OFFSET + COLONNE
1191   ;DONNE POSITION DE COLONNE ABSOLUE
1192   ;RETOUR
```

1193 FENETRE ACTUELLE HAUT, GAUCHE - HL

Cette routine convertit une position absolue en une position relative. Lors de l'appel comme après l'abandon de la routine, H contient l'indication de la colonne et L l'indication de la ligne.

```
1193   ;(FENETRE ACTUELLE HAUT)
1196   ;FENETRE ACTUELLE HAUT - LIGNE
1197   ;FORMER LE COMPLEMENT A DEUX
1199   ;AJOUTER
119A   ;DONNE LA POSITION DE LIGNE RELATIVE
119B   ;(FENETRE GAUCHE ACTUELLE)
119E   ;FENETRE ACTUELLE GAUCHE - COLONNE
119F   ;FORMER COMPLEMENT A DEUX
11A0   ;AJOUTER DEUX AU CONTENU
11A1   ;DE L'ACCUMULATEUR
```

11A2 ;DONNE LA POSITION DE COLONNE RELATIVE
11A3 ;RETOUR

11A4 INVERSER CURSEUR

Avant l'exécution de la routine VERIFIER POSITION DU CURSEUR, le curseur doit être inversé ici à l'aide de TXT DRAW/UNDRAW CURSOR.

11A4 ;TXT DRAW/UNDRAW CURSOR

11A7 VERIFIER POSITION DU CURSEUR

Lors de l'abandon de VERIFIER POSITION DU CURSEUR, le double registre HL fournit la position du curseur (H=colonne, L=ligne).

```
11A7  ;(POSITION CURSEUR ACTUELLE (ROW,COL))
11AA  ;HL A L'INTERIEUR DES LIMITES DE FENETRE
11AD  ;(POSITION CURSEUR ACTUELLE (ROW,COL))
11B0  ;HL ETAIT DANS L'INTERVALLE AUTORISE
11B1  ;POSITION DU CURSEUR SUR LA PILE
11B2  ;ROLL COUNT ACTUEL
11B5  ;B VAUT &00 OU &FF
      ;B=&00 SIGNIFIE SCROLLING VERS LE BAS
      ;B=&FF SIGNIFIE SCROLLING VERS LE HAUT
11B6  ;A=&00 OU A=&FE
11B7  ;A=&01 OU A=&FF
11B8  ;AJOUTER ROLL COUNT ACTUEL
11B9  ;NOUVEAU ROLL COUNT
11BA  ;TXT GET WINDOW
11BD  ;(PAPER ACTUEL)
11C0  ;SAUVER PAPER ACTUEL ET CARRY
11C1  ;SCR SW ROLL
11C4  ;ALLER CHERCHER PAPER ACTUEL ET CARRY
11C5  ;SCR HW ROLL
11C8  ;RETIRER POSITION DU CURSEUR DE LA PILE
11C9  ;TERMINE
```

11CA TXT VALIDATE

TXT VALIDATE détermine si le curseur figure à l'intérieur de la fenêtre de texte. Lors de l'appel comme lors de l'abandon de cette routine, le registre H contient la valeur de colonne et le registre L la valeur de ligne. Si le flag CARRY est mis lors de l'abandon de TXT VALIDATE, c'est que le curseur est situé dans la zone autorisée. Un CARRY non mis signifie Scrolling.

```
11CA   ;FENETRE ACTUELLE HAUT, GAUCHE + HL
11CD   ;HL A L'INTERIEUR DES LIMITES DE FENETRE
11D0   ;SAUVER ACCU ET FLAGS
11D1   ;FENETRE ACTUELLE HAUT, GAUCHE - HL
11D4   ;ACCU ET FLAGS COMME AVANT
11D5   ;RETOUR
```

11D6 HL A L'INTERIEUR DES LIMITES DE FENETRE

Cette routine ramène la position de colonne et ligne transmise dans le registre double HL dans les limites de la fenêtre. Si le flag CARRY est mis lors de l'abandon de HL A L'INTERIEUR DES LIMITES DE FENETRE, c'est qu'aucun scrolling de l'écran n'est nécessaire pour représenter la position prescrite. Si le CARRY n'est pas mis et si le registre B contient &FF, il faut un scrolling vers le haut. Si le registre B contient &00, il faut un scrolling vers le bas.

```
11D6   ;(TXT FENETRE ACTUELLE DROITE)
11D9   ;COLONNE PAS SUPERIEURE AU BORD DROIT
11DA   ;ALORS SAUTER
11DD   ;(TXT FENETRE ACTUELLE GAUCHE)
11E0   ;RETRANSMETTRE VALEUR DANS H
11E1   ;AUGMENTER LIGNE DE 1
11E2   ;(TXT FENETRE ACTUELLE GAUCHE)
11E5   ;DIMINUER FENETRE ACTUELLE GAUCHE
11E6   ;COLONNE SUPERIEURE A ACCUMULATEUR?
11E7   ;ALORS SAUTER
11EA   ;(TXT FENETRE ACTUELLE DROITE)
11ED   ;RETRANSMETTRE VALEUR DANS H
11EE   ;DIMINUER LIGNE DE 1
11EF   ;(TXT FENETRE ACTUELLE HAUT)
```

11F2 ;DIMINUER FENETRE ACTUELLE HAUT
11F3 ;LIGNE INFERIEURE OU EGALE A ACCU
11F4 ;CELA NE DOIT PAS ETRE! CORRIGER
11F7 ;(TXT FENETRE ACTUELLE BAS)
11FA ;LIGNE INFERIEURE OU EGALE A ACCU
11FB ;FIXER CARRY (PAS DE SCROLLING)
11FC ;A CETTE CONDITION, C'EST FINI
11FD ;TXT FENETRE ACTUELLE BAS DEVIENT LIGNE
11FE ;SCROLLING VERS LE HAUT
1200 ;ANNULER CARRY (SCROLLING)
1201 ;TERMINE

1202 CORRIGER

1202 ;RESTAURER FENETRE TXT ACTUELLE HAUT
1203 ;TXT FENETRE ACTUELLE HAUT DEVIENT LIGNE
1204 ;SCROLLING VERS LE BAS
1206 ;ANNULER CARRY (SCROLLING)
1207 ;TERMINE

1208 TXT WIN ENABLE

La taille de la fenêtre de texte actuelle est fixée ici. Avant appel de TXT WIN ENABLE, les registres doubles HL et DE doivent recevoir les valeurs fixant la taille de la fenêtre. H = limite gauche, L = limite supérieure, D = limite droite et E = limite inférieure.

1208 ;SCR CHAR LIMITS
120B ;LIMITE DROITE DANS L'ACCUMULATEUR
120C ;DEPASSEMENT DE ZONE COLONNE
120F ;VALEUR MANIPULEE DANS LIMITE GAUCHE
1210 ;LIMITE DROITE DANS ACCUMULATEUR
1211 ;DEPASSEMENT DE ZONE COLONNE
1214 ;VALEUR MANIPULEE DANS LIMITE DROITE
1215 ;LIMITES DROITE GAUCHE CORRECTES?
1216 ;SAUTER ECHANGE
1218 ;LIMITE GAUCHE DANS LIMITE DROITE
1219 ;LIMITE DROITE DANS LIMITE GAUCHE
121A ;LIMITE SUPERIEURE DANS ACCUMULATEUR

121B ;DEPASSEMENT DE ZONE LIGNE
121E ;VALEUR MANIPULEE DANS LIMITE SUPERIEURE
121F ;LIMITE INFERIEURE DANS ACCUMULATEUR
1220 ;DEPASSEMENT DE ZONE LIGNE
1223 ;VALEUR MANIPULEE DANS LIMITE INFERIEURE
1224 ;LIMITES SUPERIEURE INFERIEURE CORRECTES
1225 ;SAUTER ECHANGE
1227 ;LIMITE SUPERIEURE DANS LIMITE INFERIEURE
1228 ;LIMITE INFERIEURE DANS LIMITE SUPERIEURE
1229 ;ECRIRE LIMITE GAUCHE, LIMITE SUPERIEURE
122C ;ECRIRE LIMITE DROITE, LIMITE INFERIEURE
1230 ;LIMITE GAUCHE DANS ACCUMULATEUR
1231 ;LIMITE GAUCHE OR LIMITE SUPERIEURE
1232 ;SW ROLL
1234 ;LIMITE DROITE DANS ACCUMULATEUR
1235 ;LIMITE DROITE XOR PLUS GRANDE VALEUR AUTORISEE
1236 ;SW ROLL
1238 ;LIMITE SUPERIEURE DANS ACCUMULATEUR
1239 ;LIMITE SUPERIEURE XOR PLUS GRANDE VALEUR AUTORISEE
123A ;(TXT SCROLL FLAG)
123D ;PASSER AU POSITIONNEMENT DU CURSEUR

1240 DEPASSEMENT DE ZONE COLONNE

Les registres A et B sont utilisés pour l'entrée et la sortie. L'accumulateur contient, aussi bien en entrée qu'en sortie, la limite de colonne d'une fenêtre. Le registre B contient la plus grande valeur autorisée pour la limite de colonne.

1240 ;LIMITE DE COLONNE SUPERIEURE OU EGALE A ZERO
1241 ;SAUTER A CETTE CONDITION
1244 ;ZERO POUR LA LIMITE DE COLONNE
1245 ;LIMITE DE COLONNE INFERIEURE A LA
;PLUS GRANDE VALEUR AUTORISEE?
1246 ;C'EST LE CAS, ALORS TERMINE
1247 ;LA PLUS GRANDE VALEUR AUTORISEE
;DEVIENT LA LIMITE DE COLONNE
1248 ;TERMINE

1249 DEPASSEMENT DE ZONE LIGNE

Les registres A et C sont utilisés pour l'entrée et la sortie. L'accumulateur contient la limite de ligne d'une fenêtre aussi bien en entrée qu'en sortie. Le registre C contient la plus grande valeur autorisée pour la limite de ligne.

```
1249   ;LIMITE DE LIGNE SUPERIEURE OU EGALE A ZERO
124A   ;SAUTER A CETTE CONDITION
124D   ;ZERO POUR LA LIMITE DE LIGNE
124E   ;LIMITE DE LIGNE INFERIEURE
       ;PLUS GRANDE VALEUR AUTORISEE
124F   ;C'EST LE CAS, ALORS TERMINE
1250   ;LA PLUS GRANDE VALEUR AUTORISEE
       ;DEVIENT LA LIMITE DE LIGNE
1251   ;RETOUR
```

1252 TXT GET WINDOW

TXT GET WINDOW fournit à la demande la taille de la fenêtre de texte actuelle. Lors de l'abandon de la routine, les registres doubles HL et DE contiennent les valeurs définissant la taille de la fenêtre. H = limite gauche, L = limite supérieure, D = limite droite et E = limite inférieure.

```
1252   ;(TXT FENETRE ACTUELLE HAUT)
1255   ;(TXT FENETRE ACTUELLE HAUT)
1259   ;(TXT SCROLL FLAG)
125C   ;CARRY ANNULE POUR HW-ROLL
125E   ;C'EST TOUT!
```

125F TXT DRAW/UNDRAW CURSOR

Fixer/annuler le curseur.

```
125F   ;(FLAG CURSEUR DE TEXTE ACTUEL)
1262   ;CURSEUR NON AUTORISE OU VERROUILLE
1264   ;ALORS RETOUR
```

1265 TXT PLACE/REMOVE CURSOR

Fixer curseur sur l'écran/retirer curseur de l'écran.

```
1265   ;SAUVER LE DOUBLE REGISTRE BC
1266   ;SAUVER LE DOUBLE REGISTRE DE
1267   ;SAUVER LE DOUBLE REGISTRE HL
1268   ;VERIFIER LA POSITION DU CURSEUR
126B   ;(TXT PEN ACTUEL)
       ;B=PAPER ACTUEL
       ;C=PEN ACTUEL
126F   ;SCR CHAR INVERT
1272   ;ANCIENNE VALEUR DE HL
1273   ;ANCIENNE VALEUR DE DE
1274   ;ANCIENNE VALEUR DE BC
1275   ;C'EST TOUT
```

1276 TXT CUR ON

Autoriser le curseur (système d'exploitation).

```
1276   ;SAUVER L'ACCU ET LES FLAGS SUR LA PILE
1277   ;BIT NUMERO 1 ANNULE
1279   ;CUR ENABLE CONT'D
127C   ;RETIRER ACCU ET FLAGS DE LA PILE
127D   ;TERMINE
```

127E TXT CUR OFF

Verrouiller le curseur (système d'exploitation, prioritaire par rapport à TXT CUR ENABLE et TXT CUR DISABLE).

```
127E   ;SAUVER ACCU ET FLAGS SUR LA PILE
127F   ;BIT NUMERO 1 EST MIS
1281   ;CUR DISABLE CONT'D
1284   ;RETIRER ACCU ET FLAGS DE LA PILE
1285   ;TERMINE
```

1286 TXT CUR ENABLE

Autoriser curseur (programme d'application).

```
1286   ;BIT NUMERO 0 EST EFFACE
```

1288 CUR ENABLE CONT'D

```
1288   ;SAUVER ACCU ET FLAGS SUR LA PILE
1289   ;TXT DRAW/UNDRAW CURSOR
128C   ;RETIRER ACCU ET FLAGS DE LA PILE
128D   ;SAUVER REGISTRE DOUBLE HL
128E   ;FLAG CURSEUR TXT ACTUEL
1291   ;MANIPULATION DE BITS
1292   ;ECRIRE EN RETOUR LE FLAG CURSEUR TXT ACTUEL
1293   ;ALLER CHERCHER REGISTRE DOUBLE HL
1294   ;TXT DRAW/UNDRAW CURSOR
```

1297 TXT CUR DISABLE

Verrouiller le curseur (programme d'application).

```
1297   ;METTRE LE BIT NUMERO 0
```

1299 CUR DISABLE CONT'D

```
1299   ;SAUVER ACCU ET FLAGS SUR LA PILE
129A   ;TXT DRAW/UNDRAW CURSOR
129D   ;RETIRER ACCU ET FLAGS DE LA PILE
129E   ;SAUVER REGISTRE DOUBLE HL
129F   ;FLAG CURSEUR TXT ACTUEL
12A2   ;MANIPULATION DE BITS
12A3   ;ECRIRE EN RETOUR FLAG CURSEUR TXT ACTUEL
12A4   ;RETIRER VALEUR POUR HL DE LA PILE
12A5   ;C'EST TOUT
```

12A6 TXT SET PEN

TXT SET PEN fixe la couleur de premier plan. La valeur pour le premier plan est transmise à la routine dans l'accumulateur.

```
12A6   ;TXT PEN ACTUEL
12A9   ;PASSER A L'EXECUTION
```

12AB TXT SET PAPER

TXT SET PAPER fixe la couleur du fond. La valeur pour la couleur du fond est transmise à la routine dans l'accumulateur.

```
12AB   ;TXT PAPER ACTUEL
12AE   ;SAUVER VALEUR DE COULEUR SUR LA PILE
12AF   ;TXT DRAW/UNDRAW CURSOR
12B2   ;REPETER VALEUR DE COULEUR
12B3   ;SCR INK ENCODE
12B6   ;FIXER VALEUR DE COULEUR CALCULEE
12B7   ;TXT DRAW/UNDRAW CURSOR
```

12BA TXT GET PEN

TXT GET PEN détermine la couleur de premier plan dont le code figurera dans l'accu lors de l'abandon de la routine.

```
12BA   ;(TXT PEN ACTUEL)
12BD   ;SCR INK DECODE
```

12C0 TXT GET PAPER

TXT GET PAPER détermine la couleur de fond dont le code figurera dans l'accu lors de l'abandon de la routine.

```
12C0   ;(TXT PAPER ACTUEL)
12C3   ;SCR INK DECODE
```

12C6 TXT INVERSE

Cette routine échange la couleur de premier plan actuelle avec la couleur de fond actuelle.

```
12C6   ;TXT DRAW/UNDRAW CURSOR
12C9   ;PAPER ACTUEL/PEN ACTUEL DANS HL
12CC   ;PAPER DANS ACCU
12CD   ;PEN DANS PAPER
12CE   ;ACCU DANS PEN
12CF   ;PAPER/PEN ECHANGES
12D2   ;TXT DRAW/UNDRAW CURSOR
```

12D4 TXT GET MATRIX

Aller chercher l'adresse du modèle de points d'un caractère. Le code du caractère est transmis à TXT GET MATRIX dans l'accumulateur. L'adresse de la matrice de points est renvoyée comme résultat dans le registre HL.

```
12D4   ;PREPARER LE REGISTRE DE
12D5   ;STOCKER LE CODE DU CARACTERE DANS E
12D6   ;TXT GET M TABLE
12D9   ;SAUTER SI PAS DE MATRICE
12DB   ;PREMIER CARACTERE DE LA MATRICE DANS D
12DC   ;RENVOYER LE CARACTERE STOCKE DANS A
12DD   ;SOUSTRAIRE LE PREMIER CARACTERE DE CARACTERE
12DE   ;FORMER COMPLEMENT DU FLAG CARRY
12DF   ;SAUTER SI PAS DE CARACTERE
12E1   ;NUMERO DANS LA MATRICE
12E2   ;PASSER AU CALCUL DE L'ADRESSE
12E4   ;C'EST ICI QUE COMMENCE LE JEU DE CARACTERES DANS LA ROM
12E7   ;SAUVER ACCU ET FLAGS SUR LA PILE
12E8   ;ANNULER D
12EA   ;DE=ADRESSE DE BASE ROM-JEU DE CARACTERES H=&00
       ;ET L=NUMERO DANS LA MATRICE
12EB   ;2HL (RESULTAT EN HL)
12EC   ;4HL (RESULTAT EN HL)
12ED   ;8HL (RESULTAT EN HL)
12EE   ;AJOUTER L'ADRESSE DE BASE
12EF   ;RETIRER ACCU ET FLAGS DE LA PILE
```

12F0 ;RESTAURER DE
12F1 ;TERMINE

12F2 TXT SET MATRIX

TXT SET MATRIX intègre une matrice de caractère définie par l'utilisateur dans le jeu de caractères. Lors de l'appel de TXT SET MATRIX, le code du caractère figure dans l'accumulateur et l'adresse de la matrice de caractère correspondante dans le registre double HL.

12F2 ;LIBERER REGISTRE HL AVANT APPEL
;DE TXT GET MATRIX
12F3 ;TXT GET MATRIX
12F6 ;RETOUR SI PAS MATRICE
12F7 ;DE EST LA DESTINATION POUR LA MATRICE DE CARACTERES
;HL EST LA SOURCE
12F8 ;COMPTEUR: COPIER HUIT OCTETS
12FB ;EXECUTER OPERATION DE COPIE
12FD ;C'EST TOUT

12FE TXT SET M TABLE

Adresse de départ et premier caractère d'une matrice de points définie par l'utilisateur. Lors de l'appel de la routine TXT SET M TABLE, le pointeur sur la matrice définie par l'utilisateur figure dans le registre HL. Si le registre D contient la valeur zéro, c'est qu'il y a une matrice utilisateur. Le registre E contient la valeur du premier caractère.

Lors de l'abandon de TXT SET M TABLE, le flag CARRY est mis s'il y avait une ancienne matrice utilisateur. Le registre double HL contient l'adresse de cette ancienne matrice utilisateur et l'accumulateur contient son premier caractère.

12FE ;SAUVER LE POINTEUR SUR LA MATRICE
12FF ;D DANS A (FLAG)
1300 ;Y A-T-IL UNE MATRICE UTILISATEUR?
1301 ;LE REGISTRE D DEVIENT &00 (PAS DE MATRICE)
1303 ;SAUTER SI PAS DE MATRICE

1305 ;LE REGISTRE D DEVIENT &FF (MATRICE)
1306 ;SAUVER D (FLAG) ET E (PREMIER CARACTERE
;A L'INTERIEUR DE LA MATRICE UTILISATEUR)
1307 ;PREMIER CARACTERE DANS C
1308 ;PLACER POINTEUR SUR MATRICE DANS DE
1309 ;PREMIER CARACTERE DANS ACCUMULATEUR
130A ;TXT GET MATRIX
130D ;OCTET FORT MATRICE DANS ACCUMULATEUR
130E ;XOR AVEC OCTET FORT POINTEUR SUR MATRICE
130F ;ADRESSE SOURCE ET DESTINATION DIFFERENTES
;COPIE POSSIBLE!
1311 ;OCTET FAIBLE MATRICE DANS ACCUMULATEUR
1312 ;XOR AVEC OCTET FAIBLE POINTEUR SUR MATRICE
1313 ;ADRESSES SOURCE ET DESTINATION IDENTIQUES
1315 ;METTRE A DISPOSITION LE REGISTRE BC
1316 ;COPIER HUIT OCTETS DE LA MATRICE AVEC LDIR
1319 ;A NOUVEAU ANCIENNE VALEUR DANS LE REGISTRE BC
131A ;PROCHAIN CARACTERE
131B ;SAUTER TANT QU'IL Y A DES CARACTERES
131D ;RETIRER LE CONTENU DE DE DE LA PILE
131E ;TXT GET M TABLE
1321 ;(TXT 1ER CARACTERE DE LA MATRICE UTILISATEUR)
1325 ;ADRESSE DE LA MATRICE UTILISATEUR
1326 ;(TXT ADR. USER MATRIX)
132A ;C'EST TOUT

132B TXT GET M TABLE

TXT GET M TABLE détermine l'adresse de départ et le premier caractère d'une matrice utilisateur. Lors de l'abandon de la routine, le numéro du premier caractère de la matrice utilisateur figure dans l'accumulateur et l'adresse de la matrice utilisateur dans le registre double HL. Un flag CARRY mis indiquera par ailleurs qu'il existe une matrice utilisateur.

132B ;(TXT PREMIER CARACTERE MATRICE UTILISATEUR)
132E ;OCTET FORT DANS ACCUMULATEUR
132F ;MODIFIER LE CARRY
1330 ;OCTET FAIBLE DANS ACCUMULATEUR
1331 ;(TXT ADR. USER MATRIX)
1334 ;TERMINE

1335 TXT WR CHAR

Représenter un caractère. On indique dans l'accumulateur quel caractère doit être représenté.

```
1335    ;CARACTERE A REPRESENTER DANS B
1336    ;(TXT FLAG CURSEUR ACTUEL)
1339    ;MODIFIER LE FLAG CARRY
133A    ;RETOUR SI LE CARRY EST MIS
133B    ;SAUVER CARACTERE DANS B SUR LA PILE
133C    ;INVERSER CURSEUR
133F    ;AJOUTER 1 A LA COLONNE POUR LE CURSEUR
1340    ;(TXT POS. CURSEUR ACTUELLE (ROW, COL))
1343    ;ANCIENNE VALEUR POUR LA COLONNE CURSEUR
1344    ;CARACTERE DE LA PILE DANS L'ACCUMULATEUR
1345    ;TXT WRITE CHAR
1348    ;TXT DRAW/UNDRAW CURSOR
```

134B TXT WRITE CHAR

Ecrire un caractère sur l'écran. Pour pouvoir remplir cette tâche, la routine a besoin d'indications spécifiant quel caractère doit être écrit et en quelle position il doit l'être. Le code du caractère à représenter est transmis dans l'accumulateur. Dans le double registre HL figure la position dans laquelle le caractère doit être sorti (H=colonne, L=ligne).

```
134B    ;SAUVER LA POSITION DE CARACTERE
134C    ;TXT GET MATRIX
134F    ;ADRESSE DE LA MATRICE (NON COMPRIMEE)
1352    ;LA SAUVER SUR LA PILE
1353    ;SCR UNPACK
1356    ;ADRESSE DE LA MATRICE (NON COMPRIMEE)
1357    ;ALLER CHERCHER LA POSITION DE CARACTERE
1358    ;SCR CHAR POSITION
135B    ;OCCUPE HUIT LIGNES DE GRILLE
135D    ;SAUVER LE CONTENU DE BC (COMPTEUR)
135E    ;SAUVER LE CONTENU DE HL
135F    ;SAUVER LE COMPTEUR UNE SECONDE FOIS
1360    ;SAUVER L'ADRESSE DE LA MATRICE NON COMPRIMEE
1361    ;ET ECHANGER DANS HL
```

1362 ;ALLER CHERCHER OCTET ADRESSE DE LA MATRICE
1363 ;SAUT
1366 ;SCR NEXT BYTE
1369 ;ALLER CHERCHER OCTET DE LA MATRICE
136A ;PROCHAIN OCTET DE LA MATRICE
136B ;ANCIEN CONTENU DE BC (COMPTEUR)
136C ;TANT QU'IL Y A ENCORE DES OCTETS
136E ;ANCIEN CONTENU DE HL
136F ;SCR NEXT LINE
1372 ;ANCIEN CONTENU DE BC (COMPTEUR)
1373 ;TESTER SI ZERO
1374 ;CONTINUER SI ENCORE DES LIGNES DE GRILLE
1376 ;TERMINE

1377 SAUT

1377 ;(TXT MODE BACKGROUND ACTUEL)
137A ;SAUT A LA ROUTINE VOULUE

137B TXT SET BACK

TXT SET BACK est chargée de fixer le mode transparent. Si la valeur 0 est transmise à cette routine dans l'accumulateur, la routine sélectionne le mode de représentation ordinaire du CPC (désactivation du mode transparent). Si l'accu contient par contre un 1, la routine active le mode transparent.

137B ;ADRESSE DESACTIVER MODE TRANSPARENT
137E ;FIXER LE MODE
137F ;FIXER LE REGLAGE NORMAL
1381 ;ADRESSE ACTIVER MODE TRANSPARENT
1384 ;(TXT MODE BACKGROUND ACTUEL)
1387 ;C'EST TOUT

1388 TXT GET BACK

TXT GET BACK détermine quel mode transparent est fixé. Si le flag ZERO n'est pas mis lors de l'abandon de la routine, le mode transparent est désactivé. Un ZERO mis signale que le mode transparent est activé.

1388 ;(TXT MODE BACKGROUND ACTUEL)
138B ;OPERANDE AUXILIAIRE DANS DE
138E ;HL PEUT CONTENIR &1392 OU &13A0
;&1392+OPERANDE AUXILIAIRE=&0000
;&13A0+OPERANDE AUXILIAIRE=&000E
138F ;OCTET FORT DANS L'ACCUMULATEUR
1390 ;MODIFICATION DES FLAGS:
;ZERO=0 MODE TRANSPARENT ACTIVE
;ZERO=1 MODE TRANSPARENT DESACTIVE
1391 ;C'EST TOUT

1392 DESACTIVER LE MODE TRANSPARENT (REGLAGE NORMAL)

L'adresse écran est transmise à la routine dans le registre DE et C contient l'octet de la matrice de caractère.

1392 ;L=TXT PEN ACTUEL H=TXT PAPER ACTUEL
1395 ;A=OCTET DE LA MATRICE DE CARACTERE
1396 ;REALISER LE NEGATIF DE LA MATRICE DE CARACTERE
1397 ;FUSION AVEC LA COULEUR PAPER
1398 ;MASQUE NEGATIF DANS LE REGISTRE B
1399 ;ALLER CHERCHER OCTET DE LA MATRICE DE CARACTERE
139A ;COLORIER AVEC COULEUR PEN
139B ;FUSION AVEC MASQUE NEGATIF (COULEUR DU FOND)
139C ;FIXER TOUS LES BITS POUR NE PAS DETRUIRE
;ACCU DANS POINTS SCR (FORCE MODE)
139E ;A PARTIR D'ICI MEME TRAITEMENT

13A0 MODE TRANSPARENT ACTIVE

L'adresse écran est transmise à MODE TRANSPARENT ACTIVE dans le registre double DE. Le registre C contient l'octet de la matrice de caractère.

13A0 ;(TXT PEN ACTUEL)
13A3 ;TXT PEN ACTUEL DANS REGISTRE B
13A4 ;ECHANGER ADRESSE ECRAN DANS HL
13A5 ;SCR PIXELS (FORCE MODE)

13A8 TXT SET GRAPHIC

Cette routine détermine si la sortie de caractères doit se faire dans la position du curseur de texte ou du curseur graphique. Si un zéro est transmis à TXT SET GRAPHIC dans l'accumulateur, la sortie de caractères se fera (comme d'habitude) dans l'emplacement du curseur de texte. Si une valeur autre que zéro est transmise, la sortie se fait dans l'emplacement du curseur graphique (instruction BASIC TAG).

```
13A8   ;(TXT GRAPH CHAR WRITE MODE (0=DIS.))
13AB   ;TERMINE
```

13AC TXT RD CHAR

TXT RD CHAR lit un caractère de l'écran. Lors de l'abandon, le caractère figure dans l'accumulateur. Si un espace a été trouvé, le flag ZERO est mis. Un flag CARRY annulé indique qu'aucun caractère n'a pu être identifié.

```
13AC   ;PREPARER LE REGISTRE DOUBLE HL
13AD   ;PREPARER LE REGISTRE DOUBLE DE
13AE   ;PREPARER LE REGISTRE DOUBLE BC
13AF   ;INVERSER LE CURSEUR
13B2   ;TXT UNWRITE CHAR
13B5   ;CARACTERE ET FLAGS SUR LA PILE
13B6   ;TXT DRAW/UNDRAW CURSOR
13B9   ;RETIRER CARACTERE ET FLAGS DE LA PILE
13BA   ;ANCIEN ETAT DE BC
13BB   ;ANCIEN ETAT DE DE
13BC   ;ANCIEN ETAT DE HL
13BD   ;C'EST TOUT
```

13BE TXT UNWRITE CHAR

Lire un caractère de l'écran. La position du caractère est transmise dans le registre double HL (H=colonne, L=ligne). La routine fournit en résultat le caractère trouvé, dans l'accumulateur. Si le flag CARRY n'est pas mis, c'est que le caractère n'a pas pu être déterminé.

Un flag ZERO mis indique qu'un espace a été trouvé.

```
13BE   ;(TXT PAPER ACTUEL)
13C1   ;ADRESSE DE LA MATRICE
13C4   ;SAUVER LA POSITION DE CARACTERE
13C5   ;SAUVER L'ADRESSE DE LA MATRICE
13C6   ;SCR REPACK
13C9   ;RETIRER L'ADRESSE DE LA MATRICE DE LA PILE
13CA   ;ET SAUVEGARDER IMMEDIATEMENT A NOUVEAU
13CB   ;NOMBRE DE LIGNES DE GRILLE DANS B
13CD   ;ALLER CHERCHER OCTET DE MATRICE
13CE   ;FORMER LE COMPLEMENT
13CF   ;ECRIRE EN RETOUR LE COMPLEMENT
13D0   ;INDIQUER OCTET SUIVANT
13D1   ;SAUTER TANT QU'IL Y A ENCORE DES LIGNES DE GRILLE
13D3   ;COMPARER MATRICE
13D6   ;RETIRER ADRESSE DE MATRICE DE LA PILE
13D7   ;RETIRER POSITION DE CARACTERE DE LA PILE
13D8   ;SAUTER SI AUCUN CARACTERE N'A ETE TROUVE
13DA   ;RETOUR SI ON A TROUVE AUTRE CHOSE
       ;QU'UN ESPACE
13DB   ;(TXT PEN ACTUEL)
13DE   ;SCR REPACK
```

13E1 COMPARER MATRICE

La routine fournit en résultat le caractère dans l'accumulateur. Si le flag CARRY n'est pas mis, c'est que le caractère n'a pu être identifié. Un flag ZERO mis indique un espace.

```
13E1   ;INITIALISER COMPTEUR
13E3   ;NIVEAU DU COMPTEUR DANS L'ACCUMULATEUR
13E4   ;TXT GET MATRIX
13E7   ;ADRESSE DE LA MATRICE
13EA   ;NOMBRE DE LIGNES DE GRILLE DANS B
13EC   ;ALLER CHERCHER OCTET DE MATRICE
13ED   ;COMPARER AVEC OCTET PROPOSE
13EE   ;SAUTER SI DIFFERENCE
13F0   ;AU PROCHAIN OCTET
13F1   ;AU PROCHAIN OCTET A COMPARER
13F2   ;SAUTER TANT QU'IL Y A ENCORE DES LIGNES DE GRILLE
```

```
13F4    ;CODE DU CARACTERE DANS ACCU
13F5    ;ESPACE TROUVE (ZERO=1)
13F7    ;CARRY=1 POUR RECHERCHE REUSSIE
13F8    ;TERMINE
13F9    ;PROCHAIN CARACTERE
13FA    ;SAUTER TANT QU'IL Y A DES CARACTERES
13FC    ;CARRY=0 POUR RECHERCHE SANS SUCCES
13FD    ;TERMINE
```

13FE TXT OUTPUT

Représenter ou exécuter caractères (de commande).

Envoie le caractère dans l'accu sur la fenêtre d'écran actuelle ou l'exécute s'il s'agit d'un caractère de commande.

Notez que cette routine utilise l'indirection TXT OUT ACTION. Si vous l'avez manipulée, TXT OUTPUT utilisera aussi votre routine au lieu de la routine figurant dans la ROM.

```
13FE    ;SAUVER LE REGISTRE DOUBLE AF
13FF    ;SAUVER LE REGISTRE DOUBLE BC
1400    ;SAUVER LE REGISTRE DOUBLE DE
1401    ;SAUVER LE REGISTRE DOUBLE HL
1402    ;TXT OUT ACTION
1405    ;RETIRER L'ANCIEN CONTENU DE HL
1406    ;RETIRER L'ANCIEN CONTENU DE DE
1407    ;RETIRER L'ANCIEN CONTENU DE BC
1408    ;RETIRER L'ANCIEN CONTENU DE AF
1409    ;RETOUR
```

140A TXT OUT ACTION

Sortie d'un caractère sur l'écran ou exécution d'un code de commande. Le code du caractère est transmis à la routine dans l'accumulateur.

```
140A    ;SAUVER LE CODE DE CARACTERE DANS C
140B    ;(TXT GRAPH CHAR WRITE MODE)
140E    ;MODIFIER LE FLAG ZERO
```

```
140F    ;CODE DU CARACTERE DANS ACCU
1410    ;GR WRITE CHAR
1413    ;(TXT COMPTEUR DE CARACTERES CONTROL BUFFER)
1416    ;CONTENU DU COMPTEUR DANS LE REGISTRE B
1417    ;CONTENU DU COMPTEUR DANS L'ACCUMULATEUR
1418    ;COMPARER AVEC LONGUEUR MAXI
141A    ;VIDER LE BUFFER DE CONTROLE
141C    ;Y A-T-IL DES CARACTERES DANS LE BUFFER DE CONTROLE?
141D    ;SAUTER A CETTE CONDITION
141F    ;CODE DU CARACTERE DANS ACCU
1420    ;S'AGIT-IL D'UN CARACTERE DE COMMANDE?
1422    ;PAS DE CARACTERE DE COMMANDE ALORS TXT WR CHAR
1425    ;COMPTEUR=COMPTEUR+1
1426    ;(TXT BUFFER DE CONTROLE COMPTEUR DE CARACTERES)
1427    ;ENVOYER NIVEAU DU COMPTEUR DANS E
1428    ;ENVOYER DANS DOUBLE REGISTRE
142A    ;ADRESSE DE BASE+NIVEAU DU COMPTEUR
142B    ;SAUVER CARACTERE
142C    ;(POINTER CONTROL BUFFER)
142F    ;CARACTERE DE COMMANDE CORRESPONDANT DANS E
1430    ;ADRESSE DE LA TABLE DE SAUT DE CARACTERES DE COMMANDE
1433    ;ADRESSE=ADRESSE+NIVEAU DE COMPTEUR
1434    ;+NIVEAU DE COMPTEUR
1435    ;+NIVEAU DE COMPTEUR
1436    ;OCTET DANS POSITION CALCULEE DANS ACCU
1437    ;MASQUER LES BITS 4 A 7
1439    ;COMPARER AVEC NIVEAU DE COMPTEUR
143A    ;ENCORE PLUS DE CARACTERES!
143B    ;(TXT CURSOR FLAG)
143E    ;AND AVEC OCTET DANS POSITION CALCULEE
143F    ;BIT 7 DANS FLAG CARRY
1440    ;VIDER BUFFER DE CONTROLE
1442    ;SUR OCTET FAIBLE POUR ADRESSE CALL
1443    ;OCTET FAIBLE POUR ADRESSE CALL
1444    ;SUR OCTET FORT POUR ADRESSE CALL
1445    ;OCTET FORT POUR ADRESSE CALL
1446    ;POINTER CONTROL BUFFER
1449    ;CARACTERE DANS ACCUMULATEUR
144A    ;SIGNIFIE: CALL (DE)
144D    ;ANNULER ACCU ET FLAG
144E    ;(TXT COMPTEUR DE CARACTERE BUFFER DE CONTROLE)
1451    ;TERMINE
```

1452 TXT VDU DISABLE

Interdire l'affichage de caractères.

1452 ;BIT 7 ET BIT 0 MIS
1454 ;CUR DISABLE CONT'D
1457 ;CONTINUER SVP

1459 TXT VDU ENABLE

Des caractères peuvent être écrits sur l'écran.

1459 ;LES BITS 1 A 6 SONT MIS
145B ;CUR ENABLE CONT'D
145E ;A SUIVRE

1460 FLAG CURSEUR ACTUEL DANS ACCU

1460 ;(TXT FALG CURSEUR ACTUEL)
1463 ;DEJA TERMINE

1464 COPIER SAUTS DE CARACTERES DE COMMANDE DEFAUT

1464 ;ANNULER ACCU ET FLAGS
1465 ;(BUFFER DE CONTROLE COMPTEUR DE CARACTERES)
1468 ;SOURCE: SAUTS DEFAUT DE CARACTERES DE COMMANDE
146B ;DESTINATION: TABLE DE SAUT DE CARACTERES DE COMMANDE
146E ;COMPTEUR: NOMBRE D'OCTETS A COPIER
1471 ;EXECUTER OPERATION DE COPIE
1473 ;TERMINE

1474 SAUTS DEFAUT DE CARACTERES DE COMMANDE

1474 &80
1475 ;00 AUCUN EFFET (CHR$(0))

1477 &81
1478 ;01 TXT WR CHAR (CHR$(1))

```
147A    &80
147B    ;02 TXT CUR DISABLE (CHR$(2))

147D    &80
147E    ;03 TXT CUR ENABLE (CHR$(3))

1480    &81
1481    ;04 SCR SET MODE (CHR$(4))

1483    &81
1484    ;05 GRA WR CHAR (CHR$(5))

1486    &00
1487    ;06 TXT VDU ENABLE (CHR$(6))

1489    &80
148A    ;07 BIP (CHR$(7))

148C    &80
148D    ;08 CRSR LEFT (CHR$(8))

148F    &80
1490    ;09 CRSR RIGHT (CHR$(9))

1492    &80
1493    ;0A CRSR DOWN (CHR$(10))

1495    &80
1496    ;0B CRSR UP (CHR$(11))

1498    &80
1499    ;0C TXT CLEAR WINDOW (CHR$(12))

149B    &80
149C    ;0D CRSR SUR DEBUT DE LIGNE (CHR$(13))

149E    &81
149F    ;0E TXT SET PAPER (CHR$(14))

14A1    &81
14A2    ;0F TXT SET PEN (CHR$(15))
```

14A4	&80
14A5	;10 EFFACER CARACTERE SUR L'EMPLACEMENT DU CURSEUR ;(CHR$(16))
14A7	&80
14A8	;11 EFFACER LIGNE JUSQU'A L'EMPLACEMENT DU CURSEUR ;(CHR$(17))
14AA	&80
14AB	;12 EFFACER LIGNE A PARTIR DE L'EMPLACEMENT DU CURSEUR ;(CHR$(18))
14AD	&80
14AE	;13 EFFACER FENETRE JUSQU'A L'EMPLACEMENT DU CURSEUR ;(CHR$(19))
14B0	&80
14B1	;14 EFFACER FENETRE A PARTIR DE L'EMPLACEMENT DU CURSEUR ;2(CHR$(20))
14B3	&80
14B4	;15 TXT VDU DISABLE (CHR$(21))
14B6	&81
14B7	;16 MODE TRANSPARENT ACTIVE/DESACTIVE (CHR$(22))
14B9	&81
14BA	;17 SCR ACCESS (CHR$(23))
14BC	&80
14BD	;18 TXT INVERSE (CHR$(24))
14BF	&89
14C0	;19 INSTRUCTION SYMBOL (CHR$(25))
14C2	&84
14C3	;1A DEFINIR FENETRE (CHR$(26))
14C5	&00
14C6	;1B AUCUN EFFET (CHR$(27))

14C8	&83
14C9	;1C INSTRUCTION INK (CHR$(28))
14CB	&82
14CC	;1D INSTRUCTION BORDER (CHR$(29))
14CE	&80
14CF	;1E CRSR HOME (CHR$(30))
14D1	&82
14D2	;1F INSTRUCTION LOCATE (CHR$(31))

14D4 TXT GET CONTROLS

TXT GET CONTROLS va chercher l'adresse de la table de saut des caractères de commande. Cette adresse figure dans le registre double HL lors de l'abandon de la routine.

14D4	;TXT TABLE DE SAUT DE CARACTERES DE COMMANDE
14D7	;RETOUR

14D8 TABLE DE PARAMETRES POUR BIP (CHR$(7))

14D8	87 00 00 5A 00 00 0B 14
14E0	00

14E1 BIP (CHR$(7))

La routine BIP produit un signal sonore CHR$(7).

14E1	;SAUVER REGISTER IX
14E3	;POINTEUR SUR LA TABLE DE PARAMETRES
14E6	;SOUND QUEUE
14E9	;ALLER CHERCHER ANCIEN ETAT DE IX
14EB	;C'EST TOUT

14EC MODE TRANSPARENT ACTIVE/DESACTIVE (CHR$(22))

14EC ;ROTATION DE PARAMETRE VERS LE CARRY
14ED ;ACCU DEVIENT &00 OU &FF
14EE ;TXT SET BACK

14F1 INSTRUCTION INK (CHR$(28))

14F1 ;POINTEUR SUR LE PREMIER PARAMETRE
14F2 ;Numéro du crayon de couleur
14F3 ;POINTEUR SUR LE SECOND PARAMETRE
14F4 ;Paramètre première valeur de couleur
14F5 ;POINTEUR SUR LE TROISIEME PARAMETRE
14F6 ;Paramètre seconde valeur de couleur
14F7 ;SCR SET INK

14FA INSTRUCTION BORDER (CHR$(29))

14FA ;POINTEUR SUR LE PREMIER PARAMETRE
14FB ;PARAMETRE PREMIERE VALEUR DE COULEUR
14FC ;POINTEUR SUR LE SECOND PARAMETRE
14FD ;PARAMETRE SECONDE VALEUR DE COULEUR
14FE ;SCR SET BORDER

1501 DEFINIR FENETRE (CHR$(26))

1501 ;POINTEUR SUR LE PREMIER PARAMETRE
1502 ;LIMITE DE FENETRE DROITE/GAUCHE
1503 ;POINTEUR SUR LE SECOND PARAMETRE
1504 ;LIMITE DE FENETRE DROITE/GAUCHE
1505 ;POINTEUR SUR LE TROISIEME PARAMETRE
1506 ;LIMITE FENETRE HAUT/BAS
1507 ;POINTEUR SUR LE QUATRIEME PARAMETRE
1508 ;LIMITE FENETRE HAUT/BAS
1509 ;LIMITE FENETRE DROITE/GAUCHE
150A ;TXT WIN ENABLE

150D INSTRUCTION SYMBOL (CHR$(25))

150D ;HL INDIQUE LE CODE DU CARACTERE
150E ;CODE DU CARACTERE DANS ACCU
150F ;HL INDIQUE ADRESSE DE MATRICE
1510 ;TXT SET MATRIX

1513 CHR$(0)

1513 ;INVERSER CURSEUR
1516 ;TXT DRAW/UNDRAW CURSOR

1519 CRSR LEFT (CHR$(8))

1519 ;MARQUE POUR DIMINUER COLONNE
151C ;CELA CONTINUE

151E CRSR RIGHT (CHR$(9))

151E ;MARQUE POUR AUGMENTER COLONNE
1521 ;C'EST ICI QUE CELA CONTINUE

1523 CRSR DOWN (CHR$(10))

1523 ;MARQUE POUR AUGMENTER LIGNE
1526 ;C'EST ICI QUE CELA CONTINUE

1528 CRSR UP (CHR$(11))

1528 ;MARQUE POUR DIMINUER LIGNE
152B ;SAUVER MARQUE SUR LA PILE
152C ;INVERSER CURSEUR
152F ;RETIRER MARQUE DE LA PILE
1530 ;POSITION CURSEUR LIGNE
1531 ;MARQUE AJOUTER LIGNE
1532 ;NOUVELLE POSITION CURSEUR (ROW)
1533 ;POSITION CURSEUR COLONNE

1534 ;MARQUE AJOUTER COLONNE
1535 ;NOUVELLE POSITION CURSEUR (COL)
1536 ;ECRIRE NOUVELLE POSITION CURSEUR PUIS
;TXT DRAW/UNDRAW CURSOR

1539 CRSR HOME (CHR$(30))

1539 ;(TXT FENETRE ACTUELLE HAUT)
153C ;PASSER AU POSITIONNEMENT DU CURSEUR

153F CRSR SUR DEBUT DE LIGNE (CHR$(13))

153F ;INVERSER CURSEUR
1542 ;(TXT FENETRE ACTUELLE GAUCHE)
1545 ;C'EST LA QUE CELA CONTINUE

1547 INSTRUCTION LOCATE (CHR$(31))

1547 ;POINTEUR SUR COLONNE CURSEUR
1548 ;POSITION CURSEUR COLONNE
1549 ;POINTEUR SUR LIGNE CURSEUR
154A ;POSITION CURSEUR LIGNE
154B ;ECHANGER DANS HL POSITION REUNIE DANS DE
154C ;TXT SET CURSOR

154F TXT CLEAR WINDOW (CHR$(12))

154F ;TXT DRAW/UNDRAW CURSOR
1552 ;H=TXT FENETRE ACTUELLE GAUCHE
;L=TXT FENETRE ACTUELLE HAUT
1555 ;(TXT POSITION CURSEUR ACTUELLE (ROW, COL))
1558 ;D=TXT FENETRE ACTUELLE DROITE
;E=TXT FENETRE ACTUELLE BAS
155C ;PASSER A L'EXECUTION

155E EFFACER CARACTERE DANS POS C (CHR$(16))

155E ;INVERSER CURSEUR
1561 ;POSITION CURSEUR COLONNE DANS D
1562 ;POSITION CURSEUR LIGNE DANS E
1563 ;PASSER A L'EXECUTION

1565 VIDER FENETRE A PARTIR DE POS C (CHR$(20))

1565 ;EFFACER LIGNE A PARTIR DE POSITION CURSEUR
1568 ;H=TXT FENETRE ACTUELLE GAUCHE
;L=TXT FENETRE ACTUELLE HAUT
156B ;D=TXT FENETRE ACTUELLE DROITE
;E=TXT FENETRE ACTUELLE BAS
156F ;POSITION CURSEUR ACTUELLE DANS ACCU
1572 ;ENVOYER LIGNE CURSEUR DANS L
1573 ;DEVIENT LIGNE DE DEPART
1574 ;LIGNE DE DEPART INFERIEURE FENETRE BAS
1575 ;RETOUR A CETTE CONDITION
1576 ;C'EST ICI QUE CELA CONTINUE

1578 VIDER FENETRE JUSQU'A POS C (CHR$(19))

1578 ;EFFACER LIGNE JUSQU'A POSITION CURSEUR
157B ;(TXT FENETRE ACTUELLE HAUT)
157E ;(TXT FENETRE ACTUELLE DROITE)
1581 ;LIMITE FENETRE DROITE DANS ACCU
1582 ;(TXT POS. CURSEUR ACTUELLE (ROW,COL))
1585 ;POSITION CURSEUR (ROW) -1
1586 ;DEVIENT POSITION FINALE
1587 ;DEBUT INFERIEUR OU EGAL A POSITION FINALE?
1588 ;RETOUR A CETTE CONDITION
1589 ;(TXT PAPER ACTUEL)
158C ;SCR FILL BOX

158F EFFACER LIGNE A PARTIR DE POS C (CHR$(18))

2

158F	;INVERSER CURSEUR
1592	;POSITION CURSEUR (ROW) COMME LIGNE
1593	;(TXT FENETRE ACTUELLE DROITE)
1596	;LIMITE FENETRE DROITE DANS D
1597	;PASSER A LA SUITE

1599 EFFACER LIGNE JUSQU'A POSITION C (CHR$(17))

1599	;INVERSER CURSEUR
159C	;ENVOYER POSITION CURSEUR DANS DE
159D	;POSITION CURSEUR (ROW) COMME LIGNE
159E	;(TXT FENETRE ACTUELLE GAUCHE)
15A1	;LIMITE GAUCHE DE LA FENETRE
15A2	;C'EST LA QUE CELA CONTINUE
15A5	;TXT DRAW/UNDRAW CURSOR

9.6.4 Graphics Screen (GRA)

Ce pack sert exclusivement à la gestion de la fenêtre graphique.

Il convient de faire les remarques suivantes au sujet des indications de coordonnées exigées par les différentes routines :

Les coordonnées sont traduites en trois étapes. L'étape la plus proche de l'utilisateur est la position par rapport à l'origine des coordonnées fixées par vous (ORIGIN). Celle-ci est convertie en une position par rapport à l'origine de l'écran (en bas à gauche). Ces deux étapes sont indépendantes du mode ! La dernière étape est l'adresse physique du point. Cette étape dépend du mode actuel !

Ces trois étapes sont encore précédées d'une quatrième lorsqu'un couple de coordonnées relatives doit être converti en une position absolue par rapport à ORIGIN.

15A8 GRA INITIALISE

Initialisation complète du pack graphique.

```
15A8    ;GRA RESET
15AB    ;PEN 1, PAPER 0
15AE    ;OCTET FORT DANS ACCU
15AF    ;GRA SET PAPER
15B2    ;OCTET FAIBLE DANS ACCU
15B3    ;GRA SET PEN
15B6    ;FIXER ORIGIN SUR 0,0
15B9    ;OCTET FORT DANS REGISTRE D
15BA    ;OCTET FAIBLE DANS REGISTRE E
15BB    ;GRA SET ORIGIN
15BE    ;PLUS PETITE VALEUR DANS DE
15C1    ;PLUS GRANDE VALEUR DANS HL
15C4    ;PLUS GRANDE VALEUR SUR LA PILE
15C5    ;PLUS PETITE VALEUR SUR LA PILE
15C6    ;GRA WIN WIDTH
15C9    ;RETIRER ANCIENS CONTENUS
15CA    ;DE LA PILE
15CB    ;GRA WIN HEIGHT
```

15DE DECODE PEN AND PAPER

Cette routine fournit en sortie la valeur pour le fond dans le registre H et la valeur pour le premier plan dans le registre L.

```
15CE   ;GRA GET PAPER
15D1   ;CRAYON DE COULEUR POUR FOND DANS H
15D2   ;GRA GET PEN
15D5   ;CRAYON DE COULEUR POUR PREMIER PLAN DANS L
15D6   ;TERMINE!
```

15D7 GRA RESET

Réinitialisation du pack graphique et copie des indirections.

```
15D7   ;INITIALISE
15DA   ;RESTORE GRA INDIRECTIONS
15DD   ;MOVE (HL+3) DANS ((HL+1)), CNT=(HL)
15E0   ;9 OCTETS
15E1   ;ADRESSE DE DESTINATION

15E3   ;CODE POUR JP
15E4   ;GRA PLOT

15E6   ;CODE POUR JP
15E7   ;GRA TEST

15E9   ;CODE POUR JP
15EA   ;GRA LINE
```

15EC GRA EXTENDED INITIALISE

```
15EC   ;ACTIVER FORCE MODE
15ED   ;SCR ACCESS
15F0   ;ACTIVER FORCE MODE
15F1   ;GRA FILL
15F4   ;GENERER PARAMETRE POUR LE PREMIER POINT
15F5   ;GRA FIRST POINT
15F8   ;GRA SET MASK
```

15FB GRA MOVE RELATIVE

Déplacement vers une position relative.

```
15FB    ;ADD COORD. ACTUELLE + COORD. REL
```

15FE GRA MOVE ABSOLUTE

Déplacement vers une position absolue.

```
15FE    ;(COORDONNEE X ACTUELLE)
1602    ;(COORDONNEE Y ACTUELLE)
1605    ;TERMINE!
```

1606 GRA ASK CURSOR

Où est le curseur graphique actuel? GRA ASK CURSOR fournit comme paramètres la coordonnée Y du curseur graphique dans le registre HL et la coordonnée X dans le registre DE.

```
1606    ;(COORDONNEE X ACTUELLE)
160A    ;(COORDONNEE Y ACTUELLE)
160D    ;C'EST TOUT
```

160E GRA SET ORIGIN

Fixer l'origine des coordonnées utilisateur. La routine attend comme paramètres d'entrée la coordonnée Y dans le registre HL et la coordonnée X dans le registre DE.

```
160E    ;(X ORIGIN)
1612    ;(Y ORIGIN)
1615    ;INITIALISER REGISTRE DE
1618    ;REGISTRE HL
1619    ;INITIALISER
161A    ;GRA MOVE ABSOLUTE
```

161C GRA GET ORIGIN

Aller chercher l'origine des coordonnées utilisateur. La routine fournit comme paramètres de sortie la coordonnée Y dans le registre HL et la coordonnée X dans le registre DE.

```
161C    ;(X ORIGIN)
1620    ;(Y ORIGIN)
1623    ;C'EST DEJA FINI!
```

1624 ALLER CHERCHER POSITION DE DEPART PHYSIQUE

```
1624    ;GRA ASK CURSOR
```

1627 ALLER CHERCHER POSITION PHYSIQUE DE DESTINATION ET FIXER CURSEUR

```
1627    ;GRA MOVE ABSOLUTE
```

162A GRA CONVERT POS

```
162A    ;SAUVER COORDONNEE Y SUR LA PILE
162B    ;SCR GET MODE
162E    ;DEFINIR MODE
1630    ;GENERER MASQUE
1632    ;INITIALISER REGISTRE H
1634    ;MASQUE DANS LE REGISTRE L
1635    ;BIT SEPT MIS?
1637    ;SAUTER A CETTE CONDITION
1639    ;PREPARER ADDITION
163A    ;AJOUTER MASQUE
163B    ;RESULTAT DANS DE
163C    ;INVERSER MASQUE
163D    ;ET FUSIONNER AVEC LE REGISTRE E
163E    ;RENVOYER RESULTAT DANS LE
        ;REGISTRE E
163F    ;SAUVER MASQUE
1640    ;(X ORIGIN)
1643    ;DIVISER PAR DEUX
```

1644 ;AVEC MASQUE CALCULE
1645 ;DIVISER HL PAR DEUX
1648 ;DIVISER MASQUE PAR DEUX
1649 ;DIVISER HL PAR DEUX
164C ;ALLER CHERCHER COORDONNEE Y
164D ;ET PLACER COORDONNEE X SUR LA PILE
164E ;ALLER CHERCHER OCTET FORT DE COORDONNEE Y
164F ;OCTET FORT*2
1650 ;LIMITE DEPASSEE
1652 ;COORDONNEE Y +1
1653 ;PLACER DANS LE BON FORMAT
1655 ;(Y ORIGIN)
1658 ;AJOUTER ORIGIN Y
1659 ;ALLER CHERCHER COORDONNEE X
165A ;DIVISER Y PAR DEUX

165D ADD COORD. ACTUELLE + COORD. REL

165D ;SAUVER CONTENU DU REGISTRE HL
165E ;(COORDONNEE X ACTUELLE)
1661 ;AJOUTER COORDONNEES
1662 ;RETIRER REGISTRE DE DE LA PILE
1663 ;RESULTAT DE L'ADDITION SUR LA PILE
1664 ;(COORDONNE Y ACTUELLE)
1667 ;AJOUTER COORDONNEES
1668 ;RETIRER REGISTRE DE DE LA PILE
1669 ;TERMINE!

166A XPOS DANS LA FENETRE?

La routine attend comme paramètre d'entrée une coordonnée X dans le registre DE. Un flag CARRY mis en sortie indique que la coordonnée figure à l'intérieur des limites de fenêtre. Si le CARRY n'est pas mis, un ZERO mis et un 0 dans l'accumulateur indiquent que la coordonnée X est inférieure à la limite de la fenêtre. Si le ZERO n'est pas mis et si l'accu contient &FF, c'est que la coordonnée est supérieure à la limite de la fenêtre.

166A ;(COORDONNEE X GRA FENETRE GAUCHE)
166D ;FIXER CARRY POUR SBC

166E	;LIMITE GAUCHE DE LA FENETRE - COORD. X (CARRY)
1670	;SAUTER A CETTE CONDITION
1673	;(COORDONNEE X GRA FENETRE DROITE)
1676	;ANNULER CARRY POUR SBC
1677	;LIMITE DROITE DE LA FENETRE - COORD. X
1679	;COORDONNEE DANS LA FENETRE
167A	;RETOUR CONDITIONNEL
167B	;COORDONNE SUPERIEURE A LA LIMITE DE LA FENETRE
167D	;RETOUR
167E	;COORDONNEE INFERIEURE A LA LIMITE DE LA FENETRE
167F	;RETOUR

1680 YPOS DANS LA FENETRE?

La routine attend comme paramètres en entrée une coordonnée Y dans le registre HL. Un flag CARRY mis en sortie indique que la coordonnée figure à l'intérieur des limites de fenêtre. Si le CARRY n'est pas mis, un ZERO mis et un 0 dans l'accumulateur indiquent que la coordonnée Y est inférieure à la limite de la fenêtre. Si le ZERO n'est pas mis et si l'accu contient &FF, c'est que la coordonnée est supérieure à la limite de la fenêtre.

1680	;(COORDONNEE Y GRA FENETRE HAUT)
1683	;ANNULER CARRY POUR SBC
1684	;LIMITE FENETRE - COORDONNEE Y
1686	;SAUTER SI COORD. Y SUPERIEURE
1689	;(COORDONNEE Y GRA FENETRE BAS)
168C	;METTRE CARRY POUR SBC
168D	;LIMITE FENETRE INFERIEURE - COORD. Y (CARRY)
168F	;SAUTER A CETTE CONDITION
1692	;COORDONNEE DANS LA FENETRE
1693	;TERMINE!

1694 POS IN WINDOW

POS IN WINDOW attend comme paramètre d'entrée une coordonnée Y dans le registre HL et une coordonnée X dans le registre DE. Un CARRY mis en sortie indique que les coordonnées figurent à l'intérieur des limites de la fenêtre.

```
1694    ;ALLER CHERCHER POS DESTINATION PHYS
        ;ET FIXER CURSEUR
1697    ;COORDONNEE Y SUR LA PILE
1698    ;XPOS DANS LA FENETRE?
169B    ;RETIRER COORDONNEE Y DE LA PILE
169C    ;POSITION HORS DE LA FENETRE
169D    ;COORDONNEE X SUR LA PILE
169E    ;REGISTRE POUR YPOS DANS LA FENETRE? ECHANGER
169F    ;YPOS DANS LA FENETRE?
16A2    ;ECHANGER A NOUVEAU LES REGISTRES
16A3    ;RETIRER COORDONNEE X DE LA PILE
16A4    ;C'EST TOUT
```

16A5 GRA WIN WIDTH

Fixer les limites gauche et droite de la fenêtre graphique. Le registre HL doit contenir pour cela la limite droite de la fenêtre et le registre DE la limite gauche.

```
16A5    ;SAUVER LA LIMITE DROITE DE LA FENETRE
16A6    ;ANCIENNE LIMITE GAUCHE DE LA FENETRE
16A9    ;ALLER CHERCHER LIMITE DROITE DE LA FENETRE
16AA    ;SAUVER LIMITE GAUCHE DE LA FENETRE
16AB    ;ANCIENNE LIMITE DROITE DE LA FENETRE
16AE    ;ALLER CHERCHER LIMITE GAUCHE DE LA FENETRE
16AF    ;LIMITE GAUCHE DE FENETRE<LIMITE DROITE
16B0    ;VOIR 16AF
16B1    ;VOIR 16AF
16B2    ;VOIR 16AF
16B3    ;SAUTER SI INFERIEUR
16B5    ;ECHANGER LIMITES
16B6    ;ALLER CHERCHER LIMITE GAUCHE
16B7    ;ET MASQUER
16B9    ;RESULTAT DANS A
16BA    ;ALLER CHERCHER LIMITE DROITE
16BB    ;ET MASQUER
16BD    ;RESULTAT DANS A
16BE    ;SCR GET MODE
16C1    ;MODE 0?
16C2    ;SPLIT VALUE
16C5    ;MODE 1?
```

16C6 ;SPLIT VALUE
16C9 ;(COORDONNEE X GRA FENETRE GAUCHE)
16CD ;(COORDONNEE X GRA FENETRE DROITE)
16D0 ;C'EST TOUT

16D1 ;OCTET FORT LIMITE GAUCHE DE FENETRE DANS ACCU
16D2 ;ELIMINER FLAG C
16D3 ;HL INIT
16D6 ;NEGATIF?
16D7 ;PLUS GRANDE VALEUR
16DA ;OCTET FAIBLE LIMITE GAUCHE DE FENETRE DANS ACCU
16DB ;DETERMINER TAILLE REELLE
16DC ;ENVOYER OCTET FORT LIMITE GAUCHE DE FENETRE DANS ACCU
16DD :DETERMINER TAILLE REELLE
16DE ;TERMINE SI PAS DE DEPASSEMENT
16DF ;FIXER A NOUVEAU ANCIENNE VALEUR
16E0 ;TERMINE!

16E1 SPLIT VALUE

Les valeurs figurant en entrée dans les registres HL et DE figurent en sortie divisées par deux dans ces mêmes registres.

16E1 ;CONTENU DU REGISTRE DE
16E3 ;DIVISER PAR DEUX
16E5 ;CONTENU DU REGISTRE HL
16E7 ;DIVISER PAR DEUX
16E9 ;TERMINE!

16EA GRA WIN HEIGHT

Fixer les limites supérieure et inférieure de la fenêtre graphique.

16EA ;SAUVER VALEURS POUR LIMITE INFERIEURE DE LA FENETRE
16EB ;LIMITE FENETRE OK?
16EE ;ALLER CHERCHER LIMITE INFERIEURE DE LA FENETRE
16EF ;LIBERER HL
16F0 ;LIMITE FENETRE OK?
16F3 ;ALLER CHERCHER LIMITE INFERIEURE
16F4 ;LIMITE INFERIEURE = LIMITE SUPERIEURE

16F5 ;VOIR 16F4
16F6 ;VOIR 16F4
16F7 ;VOIR 16F4
16FA ;ECHANGER VALEURS
16FB ;(COORDONNEE Y GRA FENETRE HAUT)
16FF ;(COORDONNEE Y GRA FENETRE BAS)
1702 ;C'EST TOUT

1703 LIMITE DE FENETRE OK?

LIMITE DE FENETRE OK? a comme paramètres d'entrée/sortie une limite de fenêtre dans le sens Y.

1703 ;OCTET FORT LIMITE SUPERIEURE DE FENETRE DANS ACCU
1704 ;ANNULER CARRY
1705 ;PLUS PETITE LIMITE DE FENETRE
1708 ;LIMITE INFERIEURE FIXEE SUR &0000
1709 ;DIVISER LIMITE FENETRE PAR DEUX
170B ;DONNE LA LIMITE REELLE
170D ;PLUS GRANDE LIMITE DE FENETRE
1710 ;OCTET FAIBLE LIMITE DE FENETRE REELLE DANS ACCU
1711 ;OCTET FAIBLE OK?
1712 ;OCTET FORT LIMITE DE FENETRE REELLE DANS ACCU
1713 ;OCTET FORT OK?
1714 ;PLUS GRANDE LIMITE DE FENETRE
1715 ;A NOUVEAU ANCIENNES LIMITES DE FENETRE
1716 ;DEJA TERMINE!

1717 GRA GET W WIDTH

Limites gauche et droite de la fenêtre? Le registre HL doit contenir pour cela la limite droite et le registre DE la limite gauche de la fenêtre.

1717 ;(COORDONNEE X GRA FENETRE GAUCHE)
171B ;(COORDONNEE X GRA FENETRE DROITE)
171E ;SCR GET MODE
1721 ;MODE-1
1722 ;MODE 1?
1725 ;MODE-1

1726 ;MODE 0?
1727 ;COORDONNE X GAUCHE * 2
1728 ;+1
1729 ;HL = COORDONNEE X DROITE
172A ;COORDONNEE X DROITE * 2
172B ;HL = COORDONNEE X GAUCHE
172C ;TERMINE!

172D GRA GET W HEIGHT

Limite supérieure et inférieure de la fenêtre graphique ?

172D ;(COORDONNEE Y GRA FENETRE HAUT)
1731 ;(COORDONNEE Y GRA FENETRE BAS)
1734 ;A SUIVRE

1736 GRA CLEAR WINDOW

Vider fenêtre graphique.

1736 ;GRA GET W WIDTH
1739 ;ANNULER CARRY
173A ;CALCULER LARGEUR DE LA FENETRE
173C ;POUR LE REGISTRE HL
173D ;DIVISER VALEUR DANS HL PAR DEUX
1740 ;DIVISER VALEUR DANS HL PAR DEUX
1743 ;DIVISER OCTET FAIBLE PAR DEUX
1745 ;RESULTAT DANS LE REGISTRE B
1746 ;(COORDONNEE Y GRA FENETRE BAS)
174A ;(COORDONNEE Y GRA FENETRE HAUT)
174D ;SAUVER COORDONNEE SUR LA PILE
174E ;ANNULER CARRY POUR SBC
174F ;CALCULER HAUTEUR DE FENETRE POUR
1751 ;REGISTRE HL
1752 ;OCTET FAIBLE DANS REGISTRE C
1753 ;(COORDONNEE X GRA FENETRE GAUCHE)
1757 ;COORDONNEE Y HAUT
1758 ;SAUVER CONTENU DU REGISTRE BC SUR LA PILE
1759 ;SCR DOT POSITION
175C ;RETIRE REGISTRE DE DE LA PILE

```
175D    ;(GRA PAPER)
1760    ;GRA PAPER DANS LE REGISTRE C
1761    ;SCR FLOOD BOX
1764    ;CURSEUR GRAPHIQUE SUR POSITION DEFAUT
```

1767 GRA SET PEN

Fixer couleur d'écriture. GRA SET PEN attend dans l'accumulateur le numéro de crayon de couleur.

```
1767    ;SCR INK ENCODE
176A    ;(GRA PEN)
176D    ;TERMINE!
```

176E GRA SET PAPER

Fixer la couleur du fond. GRA SET PAPER attend dans l'accumulateur le numéro de crayon de couleur.

```
176E    ;SCR IN ENCODE
1771    ;(GRA PAPER)
1774    ;C'EST TOUT
```

1775 GRA GET PEN

Quelle couleur d'écriture? Cette routine fournit dans l'accumulateur le crayon de couleur de la couleur d'écriture.

```
1775    ;(GRA PEN)
1778    ;A SUIVRE!
```

177A GRA GET PAPER

Quelle couleur de fond? Cette routine fournit dans l'accumulateur le crayon de couleur de la couleur d'écriture.

```
177A    ;(GRA PAPER)
177D    ;SCR INK DECODE
```

1780 GRA PLOT RELATIVE

Fixer point graphique relativement à la position curseur actuelle.

1780 ;ADD COORD. ACTUELLE + COORD. REL

1783 GRA PLOT ABSOLUTE

Fixer point graphique (absolu).

1783 ;GRA PLOT

1786 GRA PLOT

GRA PLOT attend lors de l'appel les coordonnées pour fixer le point dans les registres HL et DE (HL = coordonnée Y, DE = coordonnée X).

1786 ;POS IN WINDOW
1789 ;POSITION EN DEHORS DE LA FENETRE
178A ;SCR DOT POSITION
178D ;(GRA PEN)
1790 ;TRANSFERER GRA PEN DANS LE REGISTRE B
1791 ;SCR WRITE

1794 GRA TEST RELATIVE

Point mis (relativement au curseur actuel)?

1794 ;ADD COORD. ACTUELLE + COORD. REL

1797 GRA TEST ABSOLUTE

Point mis (absolu)?

1797 ;GRA TEST

179A GRA TEST

Sort dans l'accumulateur l'INK dans la position actuelle du curseur graphique. La position du curseur graphique est déterminée par les registres doubles HL et DE (HL = coordonnée Y, DE = coordonnée X).

```
179A    ;POS IN WINDOW
179D    ;GRA GET PAPER
17A0    ;SCR DOT POSITION
17A3    ;SCR READ
```

17A6 GRA LINE RELATIVE

Tracer une ligne de la distance actuelle à la distance relative.

```
17A6    ;ADD COORD. ACTUELLE + COORD. REL
```

17A9 GRA LINE ABSOLUTE

Tracer une ligne de la position actuelle à la position absolue.

```
17A9    ;GRA LINE
```

17AC GRA SET MASK

Sauver paramètre de l'instruction BASIC MASK dans l'accu.

```
17AC    ;PARAMETRE MASK
17AF    ;TERMINE!
```

17B0 GRA FIRST POINT

Sauver paramètre de l'instruction BASIC MASK dans l'accu. Si la valeur &FF est transmise dans l'accu, il faut dessiner le premier point d'une ligne. &00 signifie ne pas dessiner le premier point.

```
17B0    ;&FF DESSINER PREMIER POINT
        ;&00 NE PAS DESSINER PREMIER POINT
17B3    ;C'EST TOUT
```

17B4 GRA LINE

17B4	;SAUVER COORDONNEE FINALE LIGNE
17B8	;SAUVER COORDONNEE FINALE LIGNE
17B9	;POSITION DESTINATION PHYS
17BC	;SAUVER REGISTRE HL
17BD	;(BUFFER DE CALCUL COORD. X)
17C0	;ANNULER FLAG CARRY
17C1	;CALCULER DIFFERENCE
17C3	;OCTETS FORTS DANS ACCUMULATEUR
17C4	;RENVOYER RESULTAT
17C7	;COORDONNEE PLUS GRANDE?
17CA	;SAUVER REGISTRE DE SUR LA PILE
17CB	;RETIRER REGISTRE HL DE LA PILE
17CC	;(BUFFER DE CALCUL COORD. Y)
17CF	;ANNULER FLAG CARRY
17D0	;CALCULER DIFFERENCE
17D2	;OCTET FORT DANS ACCUMULATEUR
17D3	;ET ECRIRE DANS LA RAM
17D6	;COORDONNEE PLUS GRANDE?
17D9	;ALLER CHERCHER COORDONNEE Y FINALE
17DA	;ANNULER CARRY POUR SBC
17DB	;CALCULER DIFFERENCE Y
17DD	;AJOUTER COORDONNEE FINALE
17DE	;DETERMINER CARRY
17DF	;ET RENVOYER
17E2	;CHARGER NOUVELLE VALEUR DANS ACCU
17E5	;DIFF. Y SUPERIEURE DIFF.X?
17E7	;CHANGE DIFF.
17E8	;CHANGE SIGN
17EB	;VOIR &17E8
17EC	;VOIR &17E8
17F0	;VOIR &17E8
17F1	;VOIR &17E8
17F2	;PREMIER POINT
17F5	;POINT = ZERO?

1940 GRA WR CHAR

Ecrire un caractère dans la position actuelle du curseur graphique.

```
1942    ;TXT GET MATRIX
1962    ;SCR DOT POSITION
1973    ;SCR NEXT BYTE
197B    ;SCR NEXT LINE
1985    ;GRA ASK CURSOR
1998    ;GRA MOVE ABSOLUTE
19AC    ;SCR DOT POSITION
19C4    ;(GRA PEN)
19CE    ;(GRA PAPER)
19D2    ;SCR WRITE
```

19D5 GRA TRANS SWITCH

```
19D5    ;PARAMETRES POUR LE MODE FOND
19D8    ;TERMINE
```

9.7 LA PUISSANCE DU LANGAGE MACHINE

Le langage machine est une vraie puissance! Aucun langage évolué ne lui arrive à la cheville pour ce qui est de la vitesse et de l'économie de place mémoire car aucun logiciel ne parvient encore à optimiser la réalisation d'une idée de programme aussi bien que peut le faire un cerveau humain. Malheureusement, la programmation en langage machine, même pour un cerveau humain, prend encore beaucoup de temps et reste très compliquée. C'est d'ailleurs certainement pour cela que les programmeurs de système sont de plus en plus nombreux à recourir aux langages évolués. Le temps de programmation d'un problème est en effet dans presque tous les cas considérablement réduit si on renonce à la programmation directe en langage machine. Toutefois même les compilateurs industriels rencontrent souvent des limites lorsqu'il s'agit de problèmes graphiques. C'est ce qui explique que de nombreux programmeurs aient recours à un mélange de langage évolué et de langage machine. Si nous examinons de plus près la structure d'un programme ainsi réalisé, nous constaterons que le langage machine sera employé chaque fois que le problème qu'il permet de résoudre n'aurait pas pu l'être ou ne l'aurait été que dans de mauvaises conditions par un langage évolué.

Cette partie de l'ouvrage vous présentera une série de routines machine que vous pourrez utiliser dans des programmes machine ou intégrer dans vos programmes à partir du niveau du BASIC. Cela signifie que même un pur programmeur en BASIC peut exploiter dans certaines limites la puissance du langage machine.

9.7.1 Les sprites sur le CPC

Les sprites (ou lutins) sont des caractères spéciaux, définis par l'utilisateur, surdimensionnés, qui se composent souvent de plusieurs couleurs et qui peuvent être projetés sur n'importe quel point d'un graphisme de grille. Chaque sprite est géré ici comme un graphisme de grille indépendant qui pourra être affiché ou masqué sur l'écran à votre guise. Les sprites jouent un rôle particulièrement important dans le domaine de la programmation de jeux car ils ne détruisent pas le fond qu'ils recouvrent et ils peuvent par contre détruire d'autres sprites en cas de collision.

En règle générale, les sprites sont générés électroniquement par un contrôleur vidéo pour ne pas gaspiller le temps de calcul du processeur principal. Ce dernier n'a plus en effet qu'à transmettre les positions de sprite actuelles au contrôleur vidéo.

Le HD 6845 de l'AMSTRAD CPC ne nous permet cependant pas, malgré ses nombreuses possibilités, la génération interactive de sprite. C'est pourquoi de nombreux possesseurs de CPC recourent pour cela aux caractères redéfinis qu'ils peuvent faire glisser sur l'écran à l'aide de l'instruction TAG (voyez l'annexe à ce sujet). Pour qu'ils ne détruisent pas le fond qu'ils recouvrent, il faut activer le mode XOR pendant la phase d'animation. Bien que cette méthode puisse fournir des résultats pleinement satisfaisants pour quelques applications, elle présente toutefois des inconvénients sérieux si on l'examine attentivement.

Comme un caractère ne peut comporter plusieurs couleurs, le coloriage des chaînes de caractères redéfinis est limité par le nombre de caractères utilisés. La vitesse d'animation diminue d'autre part considérablement lorsqu'on déplace simultanément plusieurs caractères de sorte qu'il serait difficile de réaliser un jeu riche en rebondissements à moins de se résigner à des mouvements très saccadés. Il faut également noter malheureusement que la forme véritable d'une chaîne de caractères peut être tellement bouleversée par le mode XOR qu'il sera difficile de la reconnaître.

Pour éliminer ces inconvénients, nous allons vous présenter ici un générateur de sprite logiciel. Il est écrit entièrement en langage machine et peut être commandé aussi bien par le BASIC que par des programmes machine. Le générateur de sprite peut gérer au maximum huit sprites simultanément et il occupera, y compris les données de définition de sprites, moins de 1,5 K de la mémoire principale.

Caractéristiques des sprites

La taille d'un sprite défini se limite à une matrice de 8 fois 16 points. Comme le générateur de sprite travaille systématiquement en MODE 0, chacun de ces 128 points peut recevoir une des 16 couleurs maximum possibles. Pour vous faciliter encore le travail vraiment complexe de définition d'un sprite, nous vous présenterons un peu

plus loin un éditeur de sprite très pratique. Les sprites peuvent naturellement parfaitement être définis également sans ce programme. Le lecteur intéressé pourra lire au chapitre 9.4 quel bit doit être mis ou annulé ici ou là. Nous vous conseillons malgré tout d'utiliser l'éditeur de sprite car, même en s'aidant d'une calculatrice, la définition complète d'un seul sprite dure plus longtemps qu'il n'en faut pour taper l'éditeur complet.

Les données de sprite complètes, qui représentent 64 octets par sprite doivent être placées dans une zone de la RAM définie une fois pour toutes. Cette zone peut être calculée avec la formule

```
X=&A0C8+SN*64
```

&A0C8 étant l'adresse de base et SN le numéro de sprite (0 à 7). Les sprites qui ont été ainsi définis peuvent naturellement être stockés sur disquette pour être chargés directement dans la zone de la RAM appropriée chaque fois que cela sera nécessaire. La séquence BASIC

```
SAVE"nom de fichier",B,&A0C8+SN*64,64
```

permet de stocker un sprite spécifié par SN. Pour charger un sprite ainsi stocké, une instruction LOAD simple, sans indication d'adresse, suffit.

La position actuelle d'un sprite sur l'écran doit être transmise à travers des cellules de la mémoire réservées, qui figurent à l'intérieur de notre programme de générateur de sprite, à l'aide d'instructions POKE. Cette position (origine des coordonnées) se rapporte au coin supérieur gauche des sprites. Les coordonnées X et Y définissant cette position doivent être comprises dans les intervalles de valeurs suivants:

X= 0 à 151
Y= 0 à 183

Si la valeur transmise sort de la zone de valeurs autorisée, cette valeur sera automatiquement convertie en position 0 pour éviter tout problème de fonctionnement.

Les adresses réservées, à travers lesquelles les coordonnées de sprites doivent être transmises, peuvent être calculées à l'aide des deux formules suivantes :

Adresse pour la coordonnée X = &A559+2*SN
Adresse pour la coordonnée Y = &A55A+2*SN

Les adresses &A559 et &A55A représentent les adresses de base par rapport au sprite 0. La variable SN contient à nouveau notre numéro de sprite (0 à 7) qui fournit, multiplié par deux, l'offset permettant d'atteindre chacun des huit couples de coordonnées.

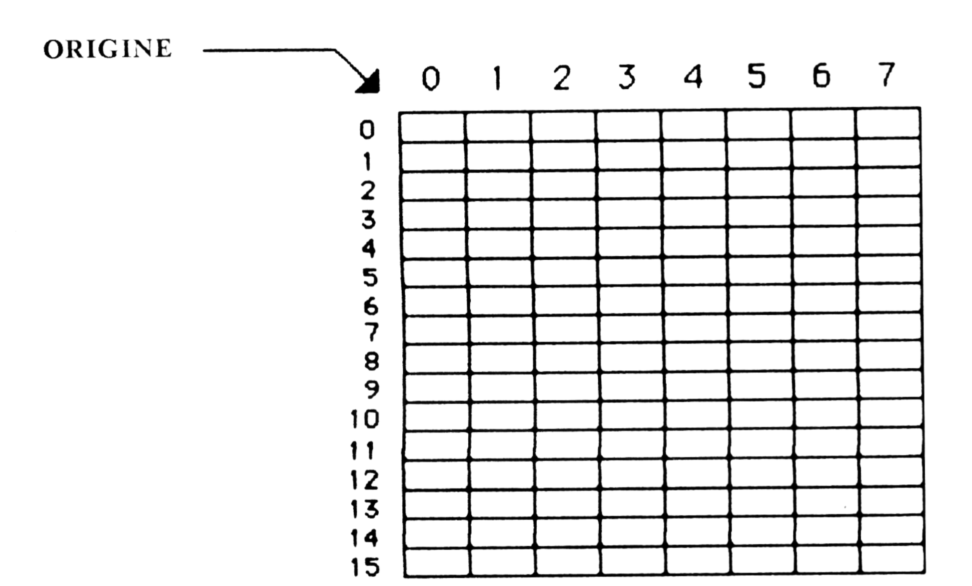


Figure 24 : **Structure de base d'un sprite**

Le listing assembleur :

```
100 ; ************************************************
110 ; **                                            **
120 ; **   GENERATEUR DE SPRITE/3.10.86 TAV         **
130 ; **                                            **
140 ; ************************************************
150 ;
160 ;
170             ORG   &A0C0
180 ;
190 FRAME:      EQU   &BD19        ;FRAME FLY BACK
200 ;
210 INIT:       LD    IX,NEWSP     ;IX=POS-COUNTER
220             CALL  GBACK        ;SAUVER FOND SPRITE
230             RET                ;RETOUR AU BASIC
240 ;
250 ;
260 ; ***** DONNEES DE SPRITE *****
270 ;
280 SPR:        DEFS 64           ;DONNEES POUR SPRITE 0
290             DEFS 64           ;DONNEES POUR SPRITE 1
300             DEFS 64           ;DONNEES POUR SPRITE 2
310             DEFS 64           ;DONNEES POUR SPRITE 3
320             DEFS 64           ;DONNEES POUR SPRITE 4
330             DEFS 64           ;DONNEES POUR SPRITE 5
340             DEFS 64           ;DONNEES POUR SPRITE 6
350             DEFS 64           ;DONNEES POUR SPRITE 7
360 ;
370 BACK:       DEFS 80           ;MEMOIRE POUR FOND SPRITE 0
380             DEFS 80           ;MEMOIRE POUR FOND SPRITE 1
390             DEFS 80           ;MEMOIRE POUR FOND SPRITE 2
400             DEFS 80           ;MEMOIRE POUR FOND SPRITE 3
410             DEFS 80           ;MEMOIRE POUR FOND SPRITE 4
420             DEFS 80           ;MEMOIRE POUR FOND SPRITE 5
430             DEFS 80           ;MEMOIRE POUR FOND SPRITE 6
440             DEFS 80           ;MEMOIRE POUR FOND SPRITE 7
450 ;
460 ONOFF:      DEFB 0            ;COMMUTATEUR SPRITE ON/OFF
470 ;
480 OLDSP:      DEFW 0            ;ANCIENNE POSITION (SPRITE 0)
```

```
490             DEFW 0          ;ANCIENNE POSITION (SPRITE 1)
500             DEFW 0          ;ANCIENNE POSITION (SPRITE 2)
510             DEFW 0          ;ANCIENNE POSITION (SPRITE 3)
520             DEFW 0          ;ANCIENNE POSITION (SPRITE 4)
530             DEFW 0          ;ANCIENNE POSITION (SPRITE 5)
540             DEFW 0          ;ANCIENNE POSITION (SPRITE 6)
550             DEFW 0          ;ANCIENNE POSITION (SPRITE 7)
560 ;
570 NEWSP:      DEFW 0          ;NOUVELLE POSITION (SPRITE 0)
580             DEFW 0          ;NOUVELLE POSITION (SPRITE 1)
590             DEFW 0          ;NOUVELLE POSITION (SPRITE 2)
600             DEFW 0          ;NOUVELLE POSITION (SPRITE 3)
610             DEFW 0          ;NOUVELLE POSITION (SPRITE 4)
620             DEFW 0          ;NOUVELLE POSITION (SPRITE 5)
630             DEFW 0          ;NOUVELLE POSITION (SPRITE 6)
640             DEFW 0          ;NOUVELLE POSITION (SPRITE 7)
650 ;
660 ;
670 ; ***** RENVOYER FOND SPRITE *****
680 ;
690 START:      CALL FRAME      ;ATTENDRE RETOUR DU FAISCEAU
700             LD   C,&08      ;INITIALISER BOUCLE
710             LD   IX,OLDSP   ;IX = POINTEUR DE POSITION
720             LD   HL,BACK    ;POINTEUR SUR LA MEMOIRE DE FOND
730 ;
740 GETB:       CALL CLCPOS     ;CALCULER OCTET DE DEPART
750 ;
760             LD   B,&10      ;LINE-COUNTER INIT
770 PLINE:      PUSH BC         ;SAUVER COMPTEUR
780             LD   BC,&0005   ;5 OCTETS PAR LIGNE
790             PUSH DE         ;SAUVER LE POINTEUR VIDEO
800             LDIR            ;RENVOYER LIGNE
810             POP  DE         ;ALLER CHERCHER POINTEUR VIDEO
820             LD   A,&08      ;DE = DE + &0800
830             ADD  A,D        ;     "
840             LD   D,A        ;     "
850             JP   NC,NOOFF   ;OFFSET DE LIGNE NEGATIF?
860             EX   DE,HL      ;HL = POINTEUR VIDEO
870             LD   BC,&3FAF   ;DECALAGE DE LIGNE POUR SBC
880             SBC  HL,BC      ;LIGNE DEFINITIVE
890             EX   DE,HL      ;HL = POINTEUR FOND
900 ;
```

```
 910 NOOFF:   POP   BC          ;Aller chercher compteur de ligne
 920          DJNZ  PLINE       ;Ligne suivante
 930 ;
 940          DEC   C           ;Fond du sprite suivant
 950          JP    NZ,GETB     ;       "
 960 ;
 970 ;
 980 ; ***** ALLER CHERCHER FOND SPRITE *****
 990 ;
1000          CALL  GBACK       ;Aller chercher fond
1010 ;
1020 ;
1030 ; ***** DESSINER SPRITES *****
1040 ;
1050          LD    C,&01       ;Init compteur de boucle
1060          LD    IX,NEWSP    ;Pointeur sur coordonnées de sprite
1070          LD    HL,SPR      ;HL = Pointeur de sprite
1080 ;
1090 PUTSPR:  LD    A,(ONOFF)   ;Sprite active?
1100          AND   C           ;       "
1110          JR    Z,NOSPR     ;       "
1120 ;
1130          CALL  CLCPOS      ;Calculer octet de départ
1140 ;
1150          LD    B,&10       ;Compteur de ligne
1160 ;
1170 DSPRLN:  PUSH  BC          ;Sauver compteur
1180          PUSH  DE          ;Sauver pointeur vidéo
1190          LD    BC,&0004    ;4 Octets par ligne
1200          LD    A,I         ;Aller chercher X
1210          AND   %00000001   ;Adresse écran paire?
1220          JR    Z,EVEN      ;       "
1230 ;
1240          LD    B,C         ;Compteur dans B
1250          LD    A,(DE)    ;Aller chercher octet de l'affichage
1260 NGSN:    AND   %10101010   ;Et masquer point droit
1270          LD    C,A         ;Résultat dans C
1280          LD    A,(HL)      ;Aller chercher données de sprite
1290          RRCA              ;Décaler point sur la droite
1300          AND   %01010101   ;Et masquer mauvais points
1310          OR    C           ;Fusion avec le fond
1320          LD    (DE),A      ;Renvoyer octet prêt
```

```
1330            INC   DE         ;Pointeur sur prochain octet
1340            LD    A,(DE)     ;Aller chercher octet de l'affichage
1350            AND   %01010101  ;Et masquer point gauche
1360            LD    C,A        ;Résultat dans C
1370            LD    A,(HL)     ;Aller chercher données de sprite
1380            RLCA             ;Décaler point sur la gauche
1390            AND   %10101010  ;Et masquer mauvais points
1400            OR    C          ;Fusion avec fond
1410            INC   HL         ;Prochaines données de sprite
1420 ;
1430            DJNZ  NGSN       ;Traiter le quartet suivant
1440            LD    (DE),A     ;Renvoyer le dernier octet
1450            JR    OFFS       ;Et aller chercher la ligne suivante
1460 ;
1470 EVEN:      LDIR             ;Copier ligne
1480 ;
1490 OFFS:      POP   DE         ;Aller chercher pointeur vidéo
1500            LD    A,&08      ;DE = DE + &0800
1510            ADD   A,D        ;     "
1520            LD    D,A        ;     "
1530            JP    NC,NOOFF2  ;Offset de ligne négatif?
1540            EX    DE,HL      ;HL = Pointeur vidéo
1550            LD    BC,&3FAF   ;Décalage de ligne pour SBC
1560            SBC   HL,BC      ;Ligne définitive
1570            EX    DE,HL      ;HL = Pointeur fond
1580 ;
1590 NOOFF2:    POP   BC         ;Aller chercher compteur
1600            DJNZ  DSPRLN     ;Ligne suivante
1610 MVEIN:     SLA   C          ;Aller chercher sprite suivant
1620            RET   Z          ;Terminé???
1630            JR    PUTSPR     ;Si ce n'est pas le cas...
1640 ;
1650 NOSPR:     LD    DE,&40     ;Sauter sprite
1660            ADD   HL,DE      ;     "
1670            INC   IX         ;     "
1680            INC   IX         ;     "
1690            JR    MVEIN      ;     "
1700 ;
1710 ;
1720 ; ***** STOCKER FOND SPRITE *****
1730 ;
1740 GBACK:     LD    C,&08      ;Initialiser boucle
```

```
1750            LD   HL,BACK     ;Pointeur sur mémoire fond
1760 ;
1770 GETB2:     CALL CLCPOS      ;Calculer octet de départ
1780 ;
1790            LD   B,&10       ;Line-Counter Init
1800            EX   DE,HL       ;HL = Pointeur vidéo
1810 GLINE:     PUSH BC          ;Sauver compteur
1820            LD   BC,&0005    ;5 Octets par ligne
1830            PUSH HL          ;Sauver compteur
1840            LDIR             ;Renvoyer ligne
1850            POP  HL          ;Aller chercher pointeur vidéo
1860            LD   BC,&0800    ;HL = HL + &0800
1870            ADD  HL,BC       ;     "
1880            JP   NC,NOOFF3   ;Offset de ligne négatif?
1890            LD   BC,&3FAF    ;Décalage de ligne pour SBC
1900            SBC  HL,BC       ;Ligne définitive
1910 ;
1920 NOOFF3:    POP  BC          ;Aller chercher compteur de ligne
1930            DJNZ GLINE       ;Ligne suivante
1940 ;
1950            EX   DE,HL       ;HL = (BACK)
1960            DEC  C           ;Prochain fond de sprite
1970            JP   NZ,GETB2    ;       "
1980 ;
1990            LD   DE,OLDSP    ;Anciennes coordonnées
2000            LD   HL,NEWSP    ;= Nouvelles coordonnées
2010            LD   BC,16       ;16 au total
2020            LDIR             ;Copier
2030 ;
2040            RET              ;C'est fait!
2050 ;
2060 ;
2070 ; ***** CALCULER ADRESSE ECRAN *****
2080 ;
2090 CLCPOS:    PUSH HL          ;Sauver HL et BC
2100            PUSH BC          ;     "
2110            LD   A,(IX+&00)  ;A = X
2120            CP   152         ;X <= 151 ?
2130            JR   C,SVRR      ;  "
2140            XOR  A           ;X = 0
2150 SVRR:      LD   C,A         ;C = X
2160            LD   I,A         ;I = X
```

```
2170 ;
2180            LD    A,(IX+&01)   ;A = Y
2190            CP    184          ;Y <= 183 ?
2200            JR    C,SVOR       ;  "
2210            XOR   A            ;X = 0
2220 SVOR:      LD    B,A          ;C = X
2230 ;
2240            AND   %00000111    ;Masquer numéro de bit
2250 ;
2260            SRL   B            ;Y = Y/8
2270            SRL   B            ;  "
2280            SRL   B            ;  "
2290 ;
2300            LD    HL,&C000     ;HL = DEBUT VIDEO
2310            JR    Z,NOMUL1     ;Multiplicateur = 0?
2320            LD    DE,80        ;HL = Y/8 * 80
2330 MUL1:      ADD   HL,DE        ;      "
2340            DJNZ  MUL1         ;      "
2350 ;
2360 NOMUL1:    AND   A            ;Multiplicateur = 0?
2370            JR    Z,NOMUL2     ;      "
2380            LD    B,A          ;Init boucle
2390            LD    DE,&0800     ;HL = HL+(Y-Y/8*8)*&800
2400 MUL2:      ADD   HL,DE        ;      "
2410            DJNZ  MUL2         ;      "
2420 ;
2430 NOMUL2:    LD    E,C          ;E = X
2440            SRL   E            ;X=X/2
2450            LD    D,&00        ;Préparer addition 16 bits
2460            ADD   HL,DE        ;HL = Adresse Vidéo
2470            EX    DE,HL        ;DE = Adresse vidéo
2480 ;
2490            INC   IX        ;Pointeur sur coordonnées suivantes
2500            INC   IX           ;  "
2510 ;
2520            POP   BC           ;Aller chercher BC et HL
2530            POP   HL           ;      "
2540            RET                ;Terminé!
2550 ;
2560            END                ;Fin du programme
```

Et maintenant, pour les amateurs de BASIC, le programme BASIC de chargement :

```
100 '*******************************************
110 '**                                       **
120 '**  PROGRAMME BASIC DE CHARGEMENT**
130 '**  POUR LE GENERATEUR DE SPRITES        **
140 '**           10.10.1986/TAV              **
150 '*******************************************
160 '
170 '
180 MEMORY &A0BF
190 FOR I=&A0C0 TO &A0C7:READ X:POKE I,X:NEXT
200 FOR I=&A0C8 TO &A568:POKE I,0:NEXT
210 FOR I=&A569 TO &A66A:READ X:POKE I,X:NEXT
220 END
230 '
240 DATA 221,33,89,165,205,247,165,201,205,25,189,14,8,221,33
250 DATA 73,165,33,200,162,205,42,166,6,16,197,1,5,0,213,237
260 DATA 176,209,62,8,130,87,210,144,165,235,1,175,63,237,66,235
270 DATA 193,16,231,13,194,117,165,205,247,165,14,1,221,33,89,165
280 DATA 33,200,160,58,72,165,161,40,68,205,42,166,6,16,197,213
290 DATA 1,4,0,237,87,230,1,40,27,65,26,230,170,79,126,15
300 DATA 230,85,177,18,19,26,230,85,79,126,7,230,170,177,35,16
310 DATA 234,18,24,2,237,176,209,62,8,130,87,210,229,165,235,1
320 DATA 175,63,237,66,235,193,16,198,203,33,200,24,182,17,64,0
330 DATA 25,221,35,221,35,24,241,14,8,33,200,162,205,42,166,6
340 DATA 16,235,197,1,5,0,229,237,176,225,1,0,8,9,210,22
350 DATA 166,1,175,63,237,66,193,16,233,235,13,194,252,165,17,73
360 DATA 165,33,89,165,1,16,0,237,176,201,229,197,221,126,0,254
370 DATA 152,56,1,175,79,237,71,221,126,1,254,184,56,1,175,71
380 DATA 230,7,203,56,203,56,203,56,33,0,192,40,6,17,80,0
390 DATA 25,16,253,167,40,7,71,17,0,8,25,16,253,89,203,59
400 DATA 22,0,25,235,221,35,221,35,193,225,201
```

Organisation et fonctionnement du générateur de sprites

Notre générateur de sprites fonctionne selon un principe simple mais très efficace. Immédiatement après le lancement de votre programme de support, il doit être initialisé une fois pour toutes par un

```
CALL &A0C0
```

C'est nécessaire pour transférer le fond des huit sprites possibles dans une zone mémoire particulière, à l'intérieur du programme de générateur. L'étape suivante consiste à copier les coordonnées de fond des huit sprites en vue d'un accès ultérieur.

Le programme de générateur de sprite est alors prêt à fonctionner. Toutefois, avant que vous ne puissiez animer vos sprites, vous devez d'abord spécifier

- combien de sprites ont été définis,
- combien de sprites doivent pouvoir être déplacés simultanément
- et quel est le rang de priorité de chaque sprite.

Les paramètres nécessaires à cet effet seront placés dans un octet unique, l'adresse &A548 de la RAM. Les bits 0 à 7 de cet octet fixent si un sprite donné est activé ou désactivé ainsi que son rang de priorité. Nous entendons par priorité la position relative d'un sprite sur un axe des Z imaginaire. Il s'agit d'un axe imaginaire car il ne concerne que les sprites et non le reste du dessin. Comme vous pouvez définir au maximum huit sprites, il y aura également huit priorités allant du niveau 0 au niveau 7. Ces niveaux de priorité correspondent au numéro de bit de votre octet de commande. Si deux sprites entrent en collision, c'est toujours le sprite de plus grand niveau de priorité qui empiétera sur celui de moindre niveau de priorité. La table suivante illustre ce principe :

Bit 0 :	(POKE &A548,&01)	activer sprite 1 (Rang de priorité 0)
Bit 1 :	(POKE &A548,&02)	activer sprite 2 (Rang de priorité 1)
Bit 2 :	(POKE &A548,&04)	activer sprite 3 (Rang de priorité 2)
Bit 3 :	(POKE &A548,&08)	activer sprite 4 (Rang de priorité 3)
Bit 4 :	(POKE &A548,&10)	activer sprite 5 (Rang de priorité 4)

Bit 5:	(POKE &A548,&20)	activer sprite 6 (Rang de priorité 5)
Bit 6:	(POKE &A548,&40)	activer sprite 7 (Rang de priorité 6)
Bit 7:	(POKE &A548,&80)	activer sprite 8 (Rang de priorité 7)

Comme le montre cette table, un bit mis correspond à un sprite activé. Si plusieurs sprites doivent être activés simultanément, les arguments de l'instruction POKE présentés ci-dessus et correspondant aux différents numéros de sprite devront être additionnés. Par exemple

```
POKE &A548,&42
```

activera les sprites numéros deux et sept. Les autres sprites resteront désactivés.

Une fois que tous les sprites ont été spécifiés de cette façon, le générateur de sprites attend dans les adresses décrites plus haut, qui sont réservées pour les coordonnées X et Y des sprites, la position actuelle du sprite à déplacer. Immédiatement après vous devez animer tous les sprites activés, c'est-à-dire les faire positionner sur les nouvelles coordonnées, avec

```
CALL &A569
```

Le programme de générateur de sprites attendra d'abord, lors de cet appel, le retour du faisceau, pour que les mouvements suivants puissent être exécutés avec le moins de tremblotements possible. Une routine reconstitue ensuite le fond qui avait été sauvé auparavant et efface ainsi tous les anciens sprites. Le fond défini par la nouvelle position actuelle transmise est alors sauvé à son tour. Les coordonnées actuelles sont en outre copiées pour permettre une reconstitution ultérieure du fond. La dernière étape consiste pour le programme à envoyer sur l'écran tous les sprites activés en fonction de leur niveau de priorité.

Pour vous montrer un exemple pratique d'utilisation du générateur de sprites, voici maintenant un petit programme de démonstration. Notez bien que le code du générateur de sprites doit se trouver en mémoire avant que vous ne lanciez ce programme.

```
100 '*************************
110 '**                     **
120 '**  DEMO SPRITES       **
130 '** 10. 10. 86/TAV      **
140 '**                     **
150 '*************************
160 '
170 '
180 MEMORY &A0BF
190 DEFINT A-Z
200 '
210 '
220 '***** SPECIFIER LES PARAMETRES ECRAN
230 '
240 MODE 0
250 BORDER 0:PAPER 5:CLS
260 INK 14,13:INK 15,11
270 '
280 '
290 '***** GENERER LES INSTRUCTIONS
300 '
310 ONOFF=&A548
320 INIT=&A0C0
330 SPRITE=&A569
340 SPRDAT=&A0C8
350 SPRPOS=&A559
360 '
370 '
380 '***** LIRE LES SPRITES
390 '
400 FOR I=SPRDAT TO SPRDAT+191
410 READ X
420 POKE I,X
430 NEXT
440 '
450 '
460 '***** INITIALISER LE GENERATEUR DE SPRITES
470 '
480 CALL INIT
490 '
500 '
510 '***** PRODUIRE LES ETOILES
```

```
520 '
530 FOR I=1 TO 100
540 PLOT(RND(1)*639),(RND(1)*399),14
550 NEXT
560 '
570 '
580 '***** PROJETER LES SPRITES
590 '
600 POKE ONOFF,3
610 Z=0
620 '
630 FOR I=0 TO 151
640 POKE SPRPOS,I:POKE SPRPOS+1,I
650 POKE SPRPOS+2,151-I:POKE SPRPOS+3,151-I
660 IF Z=2 AND I=75 THEN 750
670 CALL SPRITE
680 NEXT
690 Z=Z+1
700 GOTO 630
710 '
720 '
730 '***** EXPLOSION
740 '
750 POKE SPRPOS+4,I:POKE SPRPOS+5,I
760 POKE ONOFF,4
770 CALL SPRITE
780 FOR I=1 TO 500:NEXT
790 POKE ONOFF,0
800 CALL SPRITE
810 FOR I=1 TO 1000:NEXT
820 GOTO 600
830 '
840 '
850 '***** DONNEES SPRITE POUR LE PREMIER CHASSEUR
860 '
870 DATA 240,240,240,240
880 DATA 240,240,240,240
890 DATA 240,240,240,240
900 DATA 240,240,240,240
910 DATA 210,240,240,240
920 DATA 3,240,240,240
930 DATA 139,3,82,240
```

```
940 DATA 204,63,35,82
950 DATA 161,23,63,63
960 DATA 240,240,240,240
970 DATA 240,240,240,240
980 DATA 240,240,240,240
990 DATA 240,240,240,240
1000 DATA 240,240,240,240
1010 DATA 240,240,240,240
1020 DATA 240,240,240,240
1030 '
1040 '
1050 '***** DONNEES SPRITE POUR LE SECOND CHASSEUR
1060 '
1070 DATA 240,240,240,240
1080 DATA 240,240,240,240
1090 DATA 240,240,240,240
1100 DATA 240,240,240,240
1110 DATA 240,240,240,240
1120 DATA 240,240,240,176
1130 DATA 240,240,240,18
1140 DATA 240,165,11,101
1150 DATA 245,58,48,204
1160 DATA 60,60,41,82
1170 DATA 240,240,240,240
1180 DATA 240,240,240,240
1190 DATA 240,240,240,240
1200 DATA 240,240,240,240
1210 DATA 240,240,240,240
1220 DATA 240,240,240,240
1230 '
1240 '
1250 '***** DONNEES SPRITE POUR L'EXPLOSION
1260 '
1270 DATA 240,240,240,240
1280 DATA 240,240,240,240
1290 DATA 240,240,240,240
1300 DATA 240,216,205,240
1310 DATA 240,229,228,240
1320 DATA 228,240,218,240
1330 DATA 240,206,206,240
1340 DATA 240,156,109,240
1350 DATA 240,228,109,216
```

```
1360 DATA 240,229,218,240
1370 DATA 240,216,216,216
1380 DATA 240,240,240,240
1390 DATA 240,229,229,240
1400 DATA 240,240,240,240
1410 DATA 240,240,240,240
1420 DATA 240,240,240,240
```

Vous avez peut-être remarqué que les sprites tremblotent légèrement pendant la phase d'animation. Cela tient au fait que le temps de calcul nécessaire pour une opération de calcul de notre générateur de sprites est supérieur au délai de retour du faisceau du rayon cathodique. Pour remédier à cet inconvénient, il faut que le programme travaille avec deux pages écran indépendantes. Tous les sprites seront alors créés sur une page d'écran invisible après quoi la page d'écran visible sera échangée avec la page invisible. Nous vous indiquerons dans une section ultérieure comment programmer une telle gestion de pages d'écran.

Nota bene : Si vous possédez un CPC 464 et si vous commandez le générateur de sprites avec un programme BASIC, nous vous déconseillons d'avoir recours à une gestion d'écran en deux pages car l'organisation de la seconde page écran occupe 16 K.

L'éditeur de sprites

La définition des sprites prend beaucoup de temps, surtout en MODE 0. Pour réduire autant que possible cette perte de temps, nous allons vous présenter pour terminer ce chapitre un éditeur de sprites. Il est entièrement écrit en BASIC et il permet une définition commode et rapide d'une matrice de 8 points sur 16. Le programme contient en outre un générateur de DATA qui convertit à la demande les données du sprite terminé pour constituer des lignes de DATA qui seront sauvegardées sous forme d'un programme BASIC. Les fichiers de programme ainsi créés peuvent ensuite être réunis à votre programme principal au moyen de l'instruction MERGE.

Jeu d'instructions :

Les touches CURSEUR

Les touches curseur permettent de déplacer le curseur dans toutes les directions sur la matrice de 8 points sur 16.

Flèche haut ou flèche bas + SHIFT

Change le numéro PEN (Couleur du curseur) dans les sens croissant ou décroissant.

Touche espace (Space)

Lorsque vous actionnez la touche espace, le point sous le curseur est fixé sur la couleur activée.

Touche ","

Décale tous les points d'une position vers la gauche. Les points poussés ainsi hors de la matrice sont réinsérés sur la droite.

Touche "."

Décale tous les points d'une position sur la droite. Les points poussés ainsi hors de la matrice sont réinsérés sur la gauche.

Touche "A"

Décale tous les points d'une position vers le haut. Les points poussés ainsi hors de la matrice sont réinsérés en bas.

Touche "Z"

Décale tous les points d'une position vers le bas. Les points poussés ainsi hors de la matrice sont réinsérés en haut.

Touche "S"

Sauver sprite. Cette routine ouvre une fenêtre et vous demande d'entrer un nom de fichier. Le sprite est alors sauvé sur la disquette sous le nom entré.

Touche "L"

Charger sprite. Cette routine ouvre une fenêtre et vous demande d'entrer un nom de fichier. Le sprite est alors chargé à partir de la disquette sous le nom entré.

Touche "D"

Produire les lignes DATA. Cette routine ouvre une fenêtre et vous demande d'entrer un nom de fichier. Les données du sprite sont alors automatiquement converties en lignes DATA. Ces dernières sont stockées sur la disquette sous le nom entré, sous forme d'un programme BASIC (le programme ainsi produit peut être fusionné avec tout autre programme BASIC à l'aide de l'instruction MERGE).

Touche "N"

Nouveau - le sprite actuel est effacé.

Touche "E"

Abandon de l'éditeur de sprites.

```
100 '****************************
110 '**                        **
120 '** EDITEUR DE SPRITES     **
130 '**  8.10.86/TAV           **
140 '**                        **
150 '****************************
160 '
170 '
172 '***** INITIALISATION *****
175 '
180 MODE 0:INK 14,13:INK 15,11:PEN 1:PAPER 0:DEFINT A-Z
190 DIM P(7,15)
```

```
200 C=4:FOR I=0 TO 15:FOR J=0 TO 7:P(J,I)=5:NEXT:NEXT
210 WINDOW #1,3,17,15,18
220 SYMBOL AFTER 254
230 SYMBOL 255,0,127,127,127,127,127,127,0
240 SYMBOL 254,0,85,42,85,42,85,42,0
242 '
244 '
246 '***** CONSTRUIRE LE MASQUE ECRAN *****
248 '
250 PRINT "************************";
260 PRINT "**                    **";
270 PRINT "**";:PEN 3:PRINT" SPRITE-EDITOR  ";:PEN 1:PRINT"**";
280 PRINT "**";:PEN 3:PRINT"  8.10.86/TAV   ";:PEN 1:PRINT"**";
290 PRINT "**                    **";
300 PRINT "************************";
310 PEN 3:LOCATE 1,11:GOSUB 550:LOCATE 1,19:GOSUB 550
320 FOR I=0 TO 15:LOCATE 7,9+I:FOR J=0 TO 7:PLOT 32+J*4,94-
I*2,P(J,I):PE N P(J,I):PRINT CHR$(255);:NEXT
330 PEN 1:PRINT" P";:PEN 3:PRINT USING"##";I;:PEN I:PRINT " ";CHR$(
143);:NEXT
340 LOCATE X+7,Y+9:PEN C:PRINT CHR$(254);:LOCATE 2,12:PRINT CHR$(14
3);:LOCATE 2,13:PRINT CHR$(143);
350 LOCATE 17,C+9:PEN 4:PRINT USING"##";C;
352 '
354 '
356 '***** INTERROGATION ET DECODAGE DES INSTRUCTIONS *****
358 '
360 Y$=INKEY$:Y$=UPPER$(Y$)
370 IF Y$=CHR$(240) THEN GOSUB 540:Y=Y-1:IF Y<0 THEN Y=15
380 IF Y$=CHR$(241) THEN GOSUB 540:Y=Y+1:IF Y>15 THEN Y=0
390 IF Y$=CHR$(242) THEN GOSUB 540:X=X-1:IF X<0 THEN X=7
400 IF Y$=CHR$(243) THEN GOSUB 540:X=X+1:IF X>7 THEN X=0
410 IF Y$=" " THEN P(X,Y)=C:PLOT 32+X*4,94-Y*2,C
420 IF Y$=CHR$(244) THEN LOCATE 17,C+9:PEN 3:PRINT USING"##";C;:C=C
-1:IF  C<0 THEN C=15
430 IF Y$=CHR$(245) THEN:LOCATE 17,C+9:PEN 3:PRINT USING"##";C;:C=C
+1:IF C>15 THEN C=0
440 IF Y$="." THEN FOR J=0 TO 15:TMP=P(7,J):FOR I=6 TO 0 STEP-
1:P(I+1,J) =P(I,J):NEXT:P(0,J)=TMP:NEXT:GOTO 320
450 IF Y$="," THEN FOR J=0 TO 15:TMP=P(0,J):FOR I=1 TO 7:P(I-
1,J)=P(I,J) :NEXT:P(7,J)=TMP:NEXT:GOTO 320
460 IF Y$="A" THEN FOR I=0 TO 7:TMP=P(I,0):FOR J=1 TO 15:P(I,J-
```

```
1)=P(I,J) :NEXT:P(I,15)=TMP:NEXT:GOTO 320
470 IF Y$="Z" THEN FOR I=0 TO 7:TMP=P(I,15):FOR J=14 TO 0 STEP-
1:P(I,J+1 )=P(I,J):NEXT:P(I,0)=TMP:NEXT:GOTO 320
480 IF Y$="N" THEN RUN
490 IF Y$="S" THEN 600
500 IF Y$="L" THEN 700
510 IF Y$="D" THEN 800
520 IF Y$="E" THEN WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS:PRINT:INPUT"FI
N DU PROGRAMME";NM$:NM$=UPPER$(NM$):IF NM$="O" THEN MODE 2:PEN
1:PAPER 0:END ELSE 750
530 GOTO 340
540 LOCATE X+7,Y+9:PEN P(X,Y):PRINT CHR$(255);:RETURN
550 PRINT CHR$(150);CHR$(154);CHR$(156):PRINT CHR$(149);" ";CHR$(14
9):PRINT CHR$(149);" ";CHR$(149):PRINT CHR$(147);CHR$(154);CHR$(153
);:RETURN
560 '
570 '
580 '***** SAUVER SPRITE *****
590 '
600 WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS
610 PRINT"SAUVER SPRITE:":PRINT:INPUT"NOM";NM$
620 OPENOUT NM$
630 FOR J=0 TO 15:FOR I=0 TO 7:WRITE #9,P(I,J):NEXT:NEXT
640 CLOSEOUT
650 GOTO 750
660 '
670 '
680 '***** CHARGER SPRITE *****
690 '
700 WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS
710 PRINT"CHARGER SPRITE:":PRINT:INPUT"NOM";NM$
720 OPENIN NM$
730 FOR J=0 TO 15:FOR I=0 TO 7:INPUT #9,P(I,J):NEXT:NEXT
740 CLOSEIN
750 PAPER 0:PEN C:CLS:WINDOW SWAP 0,1:GOTO 320
760 '
770 '
780 '***** PRODUIRE ET STOCKER DATAS *****
790 '
800 WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS
810 PRINT"GENERATEUR DE DATA:":PRINT:INPUT"NOM";NM$
820 OPENOUT NM$
```

```
830 PRINT#9,"30000 'FICHIER PROGRAMME PRODUIT AUTOMATIQUEMENT"
840 Z=0:TMP=&C5F4:GOSUB 880
850 TMP=&C644:GOSUB 880
860 CLOSEOUT
870 GOTO 750
880 FOR I=0 TO 7
890 D$=STR$(30001+Z)+"DATA ":FOR J=0 TO 3
900 P$=STR$(PEEK(TMP+J+&800*I)):D$=D$+RIGHT$(P$,LEN(P$)-1)+","
910 NEXT
920 PRINT#9,LEFT$(D$,LEN(D$)-1):Z=Z+1
930 NEXT
940 RETURN
```

9.7.2 Organisation de plusieurs pages écran

Ce sous-chapitre est essentiellement destiné aux possesseurs d'un CPC 464 ou 664 car le CPC 6128 permet, avec les instructions BANK |SCREENCOPY et |SCREENSWAP, une programmation simple autant qu'efficace de plusieurs pages écran à partir du niveau du BASIC.

Le travail avec deux pages écran organisées de façon indépendante, une page visible et une page invisible, est indispensable dans certains domaines d'application. On pourra par exemple dessiner un objet quelconque sur la page écran invisible pendant que la page écran visible (actuelle) montrera sur l'écran le même objet sous une autre forme ou dans une autre position. Si on échange ensuite les pages écran, on peut obtenir un effet de dessin animé sans tremblotement. Vous trouverez d'ailleurs une application pratique de cette technique au chapitre 7.4.

Une autre application, dont l'intérêt ne doit pas être sous-estimé, consisterait à stocker deux dessins totalement différents et de structure complexe. Lorsque l'utilisateur voudra accéder au dessin invisible, le programme n'aura qu'à échanger la page visible avec la page invisible sans qu'il soit nécessaire d'effacer d'abord le dessin actuel puis d'effectuer ensuite tous les calculs nécessaires pour réaliser le second dessin voulu.

Installation

Les ordinateurs CPC 464 et CPC 664 contiennent chacun quatre banques de RAM (0 à 3). Une banque est une zone mémoire de 16 K, c'est-à-dire de même taille que la mémoire écran. C'est pourquoi la banque 3 (&C000 à &FFFF) est réservée comme mémoire écran.

Nous pourrions théoriquement définir maintenant pour notre page alternative aussi bien la banque 0, 1 ou 2. Dans la pratique, il s'avère cependant que seule la banque 1 présentera en général un intérêt (&4000 à &7FFF) car les autres banques contiennent des parties du système d'exploitation de sorte que le bon fonctionnement du système d'exploitation ne serait plus garanti si le contenu de ces emplacements de la mémoire était modifié.

Accès

En langage machine Z80, il est très facile de copier la page écran actuelle (visible) dans la page alternative (invisible) et inversement. Nous pouvons en effet utiliser l'instruction de bloc "LDIR". Les deux programmes suivants se chargent de ce travail :

```
; ***********************************************************
; **                                                       **
; **              COPY 1 TO 3                              **
; ** CE PROGRAMME COPIE LA BANQUE 1 DANS LA BANQUE 3**
; **                                                       **
; ***********************************************************
;
;
START:  LD HL,&4000        ;HL = PREMIER OCTET DE BANQUE 1
        LD DE,&C000        ;DE = PREMIER OCTET DE BANQUE 3
        LD BC,&4000        ;COPIER TOUTE LA BANQUE
;
;       LDIR               ;ET CE MAINTENANT

;       RET                ;TERMINE!
```

```
; ***********************************************************
; **                                                       **
; **               COPY 3 TO 1                             **
; ** CE PROGRAMME COPIE LA BANQUE 3 DANS LA BANQUE 1**
; **                                                       **
; ***********************************************************
;
;
START:  LD HL,&C000          ;HL = PREMIER OCTET DE LA BANQUE 3
        LD DE,&4000          ;DE = PREMIER OCTET DE LA BANQUE 1
        LD BC,&4000          ;COPIER TOUTE LA BANQUE
;
;       LDIR                 ;ET CE MAINTENANT

;       RET                  ;TERMINE!
```

Il y a par ailleurs trois possibilités pour échanger les banques 1 et 3. La première est purement logicielle, la seconde électronique et la troisième est une combinaison des deux (vecteur Jump). Chacune de ces solutions a ses avantages et sera dans certaines situations impossible, ou difficile, à remplacer par une autre.

La méthode logicielle, que nous allons vous présenter maintenant, échange les octets des deux banques réellement et non pas seulement en apparence. Cela signifie qu'après exécution de la routine suivante, le contenu complet des informations des deux banques aura été déplacé.

```
; ************************************************
; **                                            **
; **             SWITCH 1/3                     **
; **  CE PROGRAMME ECHANGE TOUS LES OCTETS **
; **         DES BANQUES 1 ET 3                 **
; **                                            **
; ************************************************
;
;
START:  LD HL,&C000          ;HL = PREMIER OCTET DE LA BANQUE 3
        LD DE,&4000          ;DE = PREMIER OCTET DE LA BANQUE 1
;
```

```
SWITCH: LD A,(HL)          ;Aller chercher octet dans banque 3
        LD B,A             ;Et stocker
        LD A,(DE)          ;Aller chercher octet dans banque 1
        LD (HL),A          ;Et le transférer dans banque 3
        LD A,B             ;Aller chercher octet stocké
        LD (DE),A          ;Et le transférer dans banque 1
;
        INC HL             ;Aller chercher octets suivants
        INC DE             ;      "
;
        LD A,H             ;Tous les octets ont été échangés?
        OR L               ;      "
        JR NZ,SWITCH       ;      "
;
;       RET                ;Terminé!
```

La solution électronique est réalisée par manipulation directe du contrôleur vidéo. Il faut pour cela placer dans le registre 12 du HD 6845 une valeur fixant à partir de quelle adresse doit commencer la zone de 16 K de la mémoire écran. La séquence d'instructions Z80

```
LD   BC,&BC00
LD   A,12
OUT  (C),A
ADD  A,4
LD   BC,&BD00
OUT  (C),A
RET
```

ou l'instruction BASIC

```
OUT &BC00,12:OUT &BD00,16
```

activent la page écran alternative (&4000) alors que

```
LD   BC,&BC00
LD   A,12
OUT  (C),A
LD   A,52
LD   BC,&BD00
OUT  (C),A
RET
```

ou l'instruction BASIC

```
OUT &BC00,12:OUT &BD00,52
```

activent à nouveau la page écran normale (&C000). Il faut noter ici que cette méthode ne consiste pas à échanger les octets des deux pages écran mais simplement à manipuler la zone d'adresses du contrôleur vidéo.

La dernière possibilité consiste à appeler une routine du système d'exploitation à laquelle le vecteur &BC06 permet d'accéder. Cette routine manipule le registre 12 du contrôleur vidéo exactement de la même façon que nous venons de vous le montrer. Toutes les routines du système d'exploitation qui ont trait au graphisme sont en outre initialisées de telle façon que la sortie graphique soit désormais détournée sur la banque définie comme page écran.

Avant appel du vecteur Jump &BC06, l'octet fort de la banque voulue (&00, &40, &80 ou &C0) doit être transmis dans l'accumulateur. Notez bien que la transmission d'autres paramètres peut entraîner l'apparition d'effets secondaires indésirables.

9.7.3 Scrolling

Le scrolling, glissement du contenu de l'écran, est une technique très appréciée essentiellement utilisée dans la programmation des jeux. Le développement de telles routines de scrolling est cependant souvent complexe et pas toujours réalisable dans des programmes exclusivement en BASIC. Le système d'exploitation de l'AMSTRAD CPC offre heureusement une série de routines de scrolling toutes prêtes qui peuvent être intégrées sans trop de difficultés dans des programmes BASIC et naturellement dans des programmes machine. Mais avant de nous intéresser de plus près à ces routines de la ROM, nous aimerions vous présenter une technique qui permet de décaler tout le contenu de l'écran, cadre compris, en un temps record.

Scrolling par impulsions HSync et VSync

Le contrôleur vidéo de l'AMSTRAD CPC contient deux registres qui déterminent le moment de l'impulsion de synchronisation horizontale ou verticale. Si cette impulsion est manipulée par l'utilisateur, la totalité de l'image sera décalée verticalement ou horizontalement.

C'est le registre deux qui est chargé du décalage horizontal et le registre sept du décalage vertical. La valeur défaut pour le registre deux est 46, celle du registre sept est 30, c'est-à-dire que si vous transmettez ces valeurs l'écran apparaît dans son format habituel. Des valeurs supérieures ou inférieures décalent l'image vers la gauche ou la droite ou bien vers le haut ou le bas.

Les deux programmes suivants présentent une application pratique de cette méthode de scrolling. Notez bien qu'ici le listing BASIC a pu être réalisé sans faire appel au langage machine.

Le listing assembleur :

```
100 ; *************************
110 ; **                     **
120 ; ** SCROLL-DEMO         **
130 ; ** 16.10.86/TAV        **
140 ; **                     **
150 ; *************************
160 ;
170 ;
180            ORG   &A000
190 ;
200 START:     LD    HL,&0207   ;Paramètres pour le contrôleur vidéo
210            LD    B,0        ;          "
220 ;
230 BEGIN:     LD    A,43       ;Valeur initiale de X
240 M1:        CALL  HMOVE      ;Déplacement horizontal
250            INC   A          ;Prochaine valeur de décalage
260            CP    50         ;Fin?
270            JR    NZ,M1      ; "
280 ;
290            LD    A,27       ;Valeur initiale de Y
300 M2:        CALL  VMOVE      ;Déplacement vertical
```

```
310          INC   A        ;Prochaine valeur de décalage
320          CP    34       ;Fin?
330          JR    NZ,M2    ; "
340 ;
350          LD    A,49     ;Valeur X finale
360 M3:      CALL  HMOVE    ;Déplacement horizontal
370          DEC   A        ;Prochaine valeur de décalage
380          CP    42       ;Fin?
390          JR    NZ,M3    ; "
400 ;
410          LD    A,33     ;Valeur Y finale
420 M4:      CALL  VMOVE    ;Déplacement vertical
430          DEC   A        ;Prochaine valeur de décalage
440          CP    26       ;Fin?
450          JR    NZ,M4    ; "
460
470          JR    BEGIN    ;Répétition
480 ;
490 HMOVE:   LD    B,&BC    ;BC = Registre d'adresse
500          OUT   (C),H    ;Charger 2 dans le registre d'adresse
510          JR    PSHDAT   ;Sortir paramètres
520 ;
530 VMOVE:   LD    B,&BC    ;BC = Registre d'adresse
540          OUT   (C),L    ;Charger 7 dans le registre d'adresse
550 ;
560 PSHDAT:  CALL  &BD19    ;Attendre retour du faisceau
570          LD    B,&BD    ;BC = Registre de données
580          OUT   (C),A    ;Charger paramètres dans
                            ;Registre de données
590          RET            ;Terminé!
600 ;
610          END            ;Fin du programme
```

Et maintenant le listing pour les amateurs de BASIC :

```
100 FOR I=43 TO 49:GOSUB 150:NEXT
110 FOR I=27 TO 33:GOSUB 160:NEXT
120 FOR I=49 TO 43 STEP-1:GOSUB 150:NEXT
130 FOR I=33 TO 27 STEP-1:GOSUB 160:NEXT
140 GOTO 100
150 CALL &BD19:OUT &BC00,2:OUT &BD00,I:RETURN
160 CALL &BD19:OUT &BC00,7:OUT &BD00,I:RETURN
170 END
```

Scrolling vertical

Le système d'exploitation nous offre pour cela une routine qui peut être appelée à travers le vecteur Jump &BC4D. La routine permet de décaler l'écran entier d'une ligne vers le haut ou le bas. La couleur de la ligne nouvellement insérée (PAPER) peut être librement définie.

La routine de scrolling attend dans le registre B un paramètre de décalage fixant la direction du décalage. Si B contient la valeur zéro, le scrolling se fera vers le bas. Tout autre valeur déplacera l'écran vers le haut. L'accumulateur doit contenir le code couleur de la nouvelle ligne. Ce code doit être défini dans le cadre du MODE activé.

Scrolling horizontal

Une routine permettant le scrolling horizontal est appelée à travers le vecteur Jump &BC05. Cette routine permet de décaler l'écran complet d'une colonne sur la gauche, la colonne décalée étant remplacée sur la droite.

La routine de décalage horizontal attend dans le registre HL un paramètre de décalage qui doit toujours être divisible par deux. Cela signifie que le bit relatif zéro du registre L doit être en état bas, c'est-à-dire valoir zéro. Autrement dit, ce paramètre doit être pair.

Scrolling de n'importe quelles portions d'image

Nous vous présenterons en dernier une routine du système d'exploitation qui décale n'importe quelles portions d'image. Cette routine peut être appelée à travers le vecteur Jump &BC50 et attend les paramètres suivants dans les registres du Z80 indiqués ci-dessous :

A: Code couleur
B: Direction de décalage
H: Colonne gauche de la zone de l'écran
L: Ligne supérieure de la zone de l'écran
D: Ligne droite de la zone de l'écran
E: Ligne inférieure de la zone de l'écran

Les limites de l'écran se réfèrent, comme d'habitude, au coin supérieur gauche de l'écran. La plus faible valeur autorisée n'est cependant pas un, comme en BASIC, mais zéro.

10. GSX - L'extension système graphique

10.1 QU'EST-CE QUE GSX ?

C'est un terme qu'on peut lire dans presque n'importe quelle revue technique ; tous les possesseurs d'un PCW le connaissent ; tout le monde aimerait pouvoir l'utiliser dans ses propres programmes mais peu de gens disposent des informations de base nécessaires pour cela. Nous voulons parler de GSX, la formule magique du graphisme sur les ordinateurs AMSTRAD PCW et CPC 6128.

Mais qu'est-ce exactement que GSX ? C'est d'abord l'abréviation de Graphics System Extension, ce qui signifie 'extension système graphique'. Elle a été développée par la société DIGITAL RESEARCH et représente sous CP/M Plus un système d'entrée/sortie graphique totalement indépendant du système employé. Les logiciels développés sous GSX sont donc extrêmement polyvalents car, entre autres,

- ils tournent sur tous les ordinateurs CPM-Plus avec extension GSX
- et ils ne prévoient pas une machine de sortie déterminée.

Remarque : Il y a des instructions graphiques qui n'auront pas les mêmes effets sur tous les périphériques de sortie. Un histogramme ne peut par exemple être sorti qu'en monochrome sur l'écran alors qu'il pourra être affiché en couleurs sur un plotter. Ces différences, qui sont fonction de la machine, ne provoquent pas de messages d'erreur. Les informations graphiques risquent donc d'être faussées dans le pire des cas mais non d'être perdues.

Cette portabilité presque illimitée permet d'obtenir une souplesse de programmation inestimable. Il n'est donc pas étonnant que GSX soit devenu un véritable standard dans le domaine des extensions graphiques pour microordinateurs.

La direction d'AMSTRAD donne malheureusement très peu d'informations sur GSX et les possibilités graphiques qu'il offre sur votre PCW ou sur le CPC 6128. Cela ne troublera certainement pas outre mesure la plupart des possesseurs d'un 6128 car le BASIC AMSTRAD dispose d'une richesse de fonctions graphiques impressionnante. Il en va tout autrement en ce qui concerne les possesseurs d'un PCW. De nombreuses revues techniques vantent en effet sa résolution graphique exceptionnelle mais l'utilisateur moyen d'un PCW n'a jamais eu l'occasion de s'en rendre compte par lui-même. Etes-vous dans ce cas? Alors vous pouvez vous rassurer car GSX peut être intégré aussi bien dans vos propres programmes machine que dans le BASIC Mallard 80. Comment? Eh bien sachez que GSX a une structure modulaire, comme CP/M. Il comporte par exemple les modules de programme GDOS (Graphic Device Operating System) et GIOS (Graphic Input Output System). Ce dernier élément correspond à un processeur de périphérique pour l'affichage vidéo, l'imprimante matricielle ou le plotter.

GDOS est chargé par contre du décodage et de la retransmission des instructions GSX au processeur de périphérique. Ses trois fonctions principales sont :

- l'organisation des appels GSX
- le chargement des processeurs de périphérique
- la conversion des paramètres en fonction du processeur

La structure interne de CP/M et la structure modulaire de GSX, qui en résultent, permettent de compléter avec GSX beaucoup de programmes écrits sous CP/M Plus. La conversion de logiciels existants est aussi simple à réaliser que possible car les disquettes système livrées avec la machine comportent un programme spécial se chargeant de ce travail. Nous vous expliquerons un peu plus tard comment cela fonctionne exactement.

Nous allons maintenant vous présenter tous les fichiers système GSX livrés avec la machine. Ils figurent sur les faces deux, trois et quatre de vos disquettes système.

Sur le PCW ainsi que sur le CPC 6128 vous avez besoin des programmes :

GSX.SYS — Ce programme contient GDOS, c'est-à-dire la véritable extension graphique.

ASSIGN.SYS — contient jusqu'à 5 noms de processeurs de périphérique avec leurs numéros de périphérique logiques. Ce fichier est stocké en format ASCII et vous pouvez donc le manipuler et le modifier à votre guise.

GENGRAF.COM — Un programme qui reconfigure le programme existant de façon à ce que GSX soit chargé automatiquement avec lui.

DDHP7470.PRL — Processeur de périphérique qui permet de connecter un plotter compatible HP sur votre PCW.

Les modules GSX suivants ne peuvent être utilisés que sur le PCW :

DDSCREEN.PRL — Processeur de périphérique chargé de la gestion de l'affichage.

DDFXLR8.PRL — Processeur de périphérique basse résolution pour l'imprimante matricielle livrée avec le PCW.

DDFXHR8.PRL — Processeur de périphérique haute résolution pour l'imprimante matricielle livrée avec le PCW.

Les programmes suivants ne figurent que sur le CPC 6128 :

DRIVERS.GSX — Fichier "Lisez-moi" dont le contenu peut être affiché à l'aide de l'instruction TYPE et qui contient les noms de tous les processeurs de périphérique livrés avec la machine.

DDFXLR7.PRL — Processeur de périphérique basse résolution pour imprimante matricielle Epson et compatible Epson.

DD-DMP1.PRL	Processeur de périphérique pour l'imprimante DMP 1 AMSTRAD.
DDSHINWA.PRL	Processeur de périphérique pour toutes les imprimantes fonctionnant avec un mécanisme Shinwa.
DDMODE0.PRL	Processeur de périphérique MODE 0 pour l'écran.
DDMODE1.PRL	Processeur de périphérique MODE 1 pour l'écran.
DDMODE2.PRL	Processeur de périphérique MODE 2 pour l'écran.

10.2 IMPLANTATION DE GSX

L'implantation de GSX est considérablement facilitée par la structure modulaire décrite au début de ce chapitre. Nous allons vous montrer maintenant à titre d'exemple comment intégrer GSX dans le BASIC Mallard 80. Ce n'est pas un hasard si nous avons choisi ce programme car tout possesseur d'un PCW possède ce BASIC et l'installation de GSX est la condition sine qua non de tout appel de GSX à partir du niveau du BASIC.

Implantation de GSX en liaison avec le BASIC Mallard

1. Formatez une disquette vierge.

2. Faites sur cette disquette une copie du BASIC Mallard.

3. Copiez en plus les programmes GSX.SYS, ASSIGN.SYS, GENGRAF.COM et au moins un processeur de périphérique. Les processeurs de périphérique sont marqués par l'extension .PRL.)

4. Entrez maintenant au clavier l'instruction : GENGRAF. Cette instruction implantera GSX automatiquement à la suite du programme figurant après GENGRAF (BASIC en l'occurrence). Le programme GENGRAF.COM peut maintenant être supprimé de votre copie car il ne sert qu'à l'implantation de GSX et n'est pas utilisé lors de l'exécution en temps réel du programme.

Tous les processeurs de périphérique que vous utilisez doivent maintenant être déclarés dans le fichier système ASSIGN.SYS. La modification du fichier est relativement simple à effectuer car il s'agit d'un fichier en format ASCII. Vous pouvez par exemple l'éditer avec le programme RPED fourni avec la machine. L'instruction

```
TYPE ASSIGN.SYS
```

affiche directement sur l'écran la liste des processeurs déjà déclarés. Le fichier ASSIGN.SYS a été intelligemment prédéfini par AMSTRAD. Il contient déjà tous les processeurs de périphérique fournis avec la machine. Le format prédéfini se présente ainsi :

```
21  à:DDFXHR8
22  à:DDFXLR8
11  à:DDHP7470
01  à:DDSCREEN
```

Chaque ligne commence par un numéro de périphérique logique de deux chiffres. Ce numéro permet d'activer le processeur correspondant à partir d'un programme d'application. On utilise normalement les numéros de périphérique

01 à 10 pour la vidéo
11 à 20 pour le plotter
et 21 à 30 pour l'imprimante

Pour préserver la compatibilité, il est préférable de respecter ces normes. Si seul un périphérique est déclaré pour une catégorie donnée, on choisira en général comme numéro le numéro le plus bas parmi ceux réservés pour cette catégorie. Le numéro de périphérique est suivi du nom du lecteur (suivi d'un double point) sur lequel figure le processeur de périphérique à charger. Le nom de lecteur peut aussi être remplacé par "a" accent grave qui indique à GDOS de chercher le processeur de périphérique voulu sur tous les lecteurs déclarés sous CP/M. Le processeur sera chargé dès qu'il aura été trouvé, s'il a été trouvé.

Si vous modifiez le fichier de configuration, notez bien que les processeurs de périphérique doivent être rangés dans ASSIGN.SYS par ordre décroissant de taille.

L'instruction CP/M Plus

```
dir d:nom de fichier.prl [size]
```

permet de connaître la taille des différents fichiers. Un fichier ASSIGN.SYS définitif ne doit pas comporter plus de cinq processeurs de périphérique.

Voilà, l'implantation de GSX est terminée. Revenez à CP/M et lancez notre nouveau BASIC intégrant GSX en entrant au clavier, comme d'habitude

```
A>BASIC
```

Confirmez votre entrée, comme toujours, avec la touche <RETURN>. Le BASIC Mallard se présentera alors avec le message :

```
--------------------------------------------------
GSX-80 1.1  01 Oct 83    Serial No 5000-1232-654321
Copyright (C) 1983
Digital Research, Inc.          All rights reserved
--------------------------------------------------

Mallard-80  BASIC with Jetsam   Version 1.29
(c) Copyright 1984  Locomotive Software Ltd
All rights reserved

17255 free bytes

Ok
```

Peut-être avez-vous noté qu'après l'implantation de GSX nous ne disposons plus que de 17255 octets (avec le fichier ASSIGN.SYS d'origine) au lieu des 31597 habituels. Que s'est-il passé? GENGRAF.COM a produit un fichier temporaire contenant un petit programme de chargement de GSX et BASIC.COM. Le BASIC Mallard a ensuite été effacé et le fichier BASIC.COM a changé de nom.

Si vous appelez désormais BASIC.COM, le programme de chargement de GSX sera d'abord transféré dans la mémoire et lancé. Il chargera alors le premier processeur de périphérique désigné dans ASSIGN.SYS dans la zone haute de la RAM. Il placera ensuite le programme GSX.SYS immédiatement devant le processeur, implantant ainsi GDOS. L'adresse de fonction du BDOS &0005 (hexa) sera à cet effet détournée de façon à ce que tous les appels système soient obligés de passer par GDOS où on testera, lors de chaque appel de fonction, s'il s'agit d'une instruction GSX ou d'une instruction BDOS. Dans ce dernier cas, GSX sautera sans autre détour au BDOS normal. Finalement le programme de chargement de GSX transfère BASIC.COM dans la zone mémoire normale (&0100 hexa).

Comme nous le montre nettement la figure 25, page suivante, GSX.SYS est un module de programme relogeable, c'est-à-dire qu'il ne doit pas forcément être chargé dans une zone de la RAM bien définie. Comme GSX, de même que tous les programmes CP/M commerciaux, est écrit en langage machine 8080, langage qui n'autorise pas les sauts relatifs, on se sert ici d'une petite astuce. Le chargeur de GSX comporte en effet une petite routine qui manipule le code 8080 de façon à ce que GSX.SYS puisse être chargé dans n'importe quelle zone mémoire dont l'octet faible de l'adresse initiale soit &00. Un offset (décalage) approprié est à cet effet ajouté aux octets forts des adresses de destination de toutes les instructions de chargement, de saut et d'appel. Les octets faibles ne sont pas affectés par cette opération. La "relogeabilité" ainsi obtenue est très importante car la taille du processeur de périphérique placé directement après GSX.SYS ne sera pas toujours forcément identique.

GSX ne permet pas, par ailleurs, de gérer plusieurs processeurs de périphérique à la fois. Si plusieurs périphériques d'entrée/sortie doivent être commandés, le logiciel devra chaque fois charger le processeur nécessaire pendant le déroulement du programme. Comme la phase d'initialisation place en mémoire le premier processeur de périphérique déclaré dans ASSIGN.SYS, GSX réserve une zone de la RAM dont la taille corresponde à ce processeur.

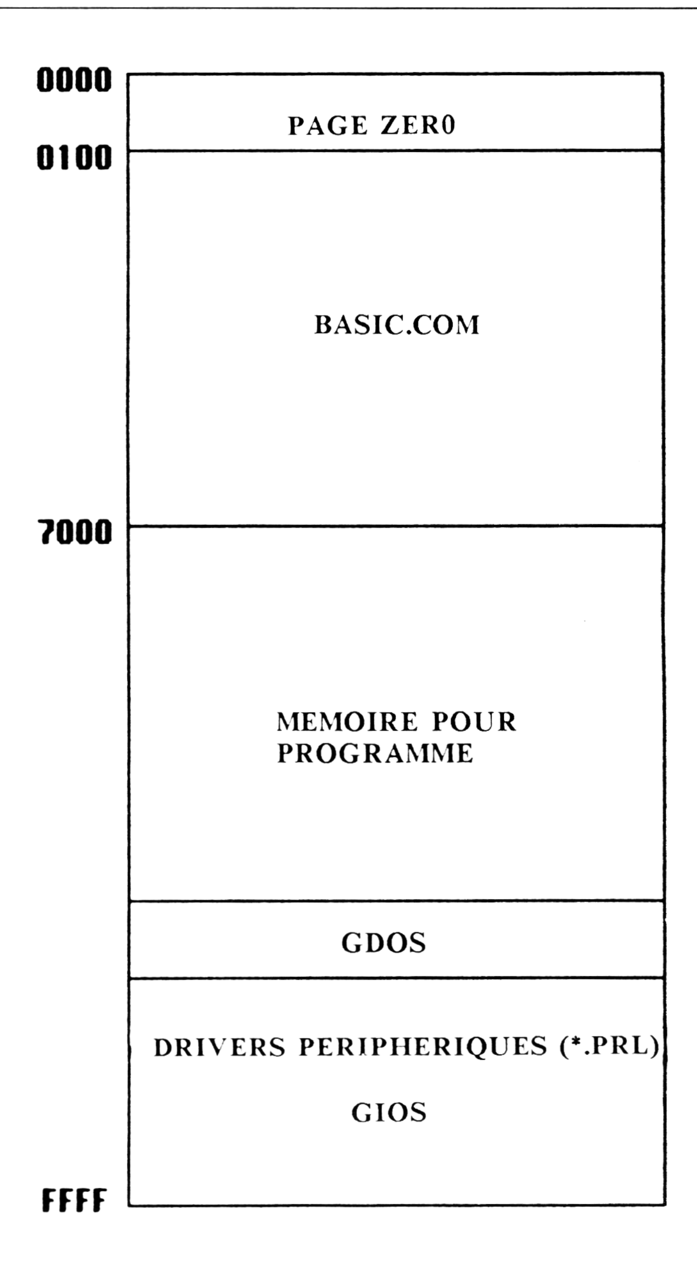


Figure 25 : Organisation de la mémoire sous GSX

C'est cela qui explique que les noms de processeurs figurant dans ASSIGN.SYS doivent être triés par ordre de taille décroissant. Il est évident en effet que le chargement en cours de programme d'un processeur plus grand que la place mémoire réservée pour lui empiéterait sur une partie de la zone système qui serait ainsi détruite.

La conséquence probable en serait un plantage définitif et irrémédiable du système.

10.3 TRANSMISSION DE PARAMETRES

Toutes les routines GSX sont appelées à travers l'adresse de page zéro modifiée &0005 (hexa). GDOS attend toujours le mode de commande 115 (&73 hexa) dans le registre C car toutes les autres valeurs sont traitées comme instructions BDOS. Dans le registre DE devra figurer la valeur initiale, sur 16 bits, d'un bloc de données de 10 octets. Ces données représentent à leur tour cinq adresses de départ des véritables blocs de paramètres dont GSX a besoin.

Ces règles un peu compliquées sont explicitées par la table suivante :

GSX-CALL	**&0005**
C	Mot de commande GDOS (&73)
DE	Adresse initiale d'un bloc de données de 10 octets
(DE)+&00	Adresse initiale du bloc de paramètres CONTROL
(DE)+&02	Adresse initiale du bloc de paramètres SETPAR
(DE)+&04	Adresse initiale du bloc de paramètres SETPOS
(DE)+&06	Adresse initiale du bloc de paramètres GETPAR
(DE)+&08	Adresse initiale du bloc de paramètres GETPOS

Si nous examinons cette table de plus près, nous y reconnaissons des registres d'entrée et de sortie d'une même sorte de variables. Seuls les registres d'entrée sont utilisés pour la transmission de paramètres. Les registres de sortie sont modifiés par les routines GSX. Leur valeur peut être réutilisée dans d'autres calculs après un appel de GSX.

Le bloc de paramètres CONTROL est chargé de la gestion interne de toutes les routines GSX. CONTROL 0, 1 et 3 attendent des paramètres que vous devez définir vous-même. CONTROL 2, 4 et 5 fournissent par contre des informations importantes après un appel de GSX.

CONTROL(0) contient le code de fonction sélectionnant la routine GSX à appeler.

CONTROL(1) contient le nombre de points auxquels des coordonnées ont été affectées.

CONTROL(2) est modifié par GSX et contient après appel d'une routine graphique le nombre de paramètres transmis à l'utilisateur dans GETPAR.

CONTROL(3) contient le nombre de paramètres devant être transmis dans SETPAR à GSX.

CONTROL(4) est modifié par GSX et contient après appel d'une routine graphique le nombre de points dont les coordonnées sont renvoyées dans GETPOS.

CONTROL(5) est modifié par GSX et contient une valeur dont la signification varie d'une routine à l'autre.

Les registres SETPAR et GETPAR contiennent des informations déterminant comment la fonction GSX sélectionnée doit réagir et comment elle a effectivement réagi.

SETPOS et GETPOS concernent la spécification d'une fonction dans le système de coordonnées. SETPOS attend la transmission de coordonnées sous la forme x1, y1, x2, y2 ... GETPOS renvoie les coordonnées importantes à l'utilisateur dans le même ordre. Comme GSX est une extension graphique universelle pour ordinateurs, et que, de ce fait, les paramètres d'entrée/sortie ne sont pas définis pour un périphérique de sortie déterminé ni pour un nombre de périphériques de sortie déterminé, GSX attend toutes les coordonnées dans une zone de définition de valeur universelle. Cette zone comprend les valeurs &0000 à &7FFF (0 à 32767).

Pour calculer une coordonnée universelle de ce type, le programmeur doit multiplier les positions X et Y de la coordonnée voulue par une constante. Ces constantes sont elles-mêmes calculées en divisant chaque fois 32767 par les résolutions maximum X et Y du périphérique de sortie concerné.

Chaque élément du bloc de paramètres est stocké sous la forme d'une variable entière sur deux octets. Cette organisation adaptée des données nous facilite heureusement la transmission des paramètres en BASIC car les paramètres nécessaires peuvent être affectés à des variables entières ordinaires.

Remarque : La description qui suit maintenant présente une possibilité de transmission de paramètres à partir du niveau du BASIC. Nous conseillons toutefois aux purs programmeurs Z80 de ne pas négliger cette section car elle contient de nombreuses informations précieuses.

En BASIC vous pouvez définir des variables numériques normales ou bien des variables dites entières. Ces dernières présentent l'avantage que le système d'exploitation BASIC ne réserve que deux octets en mémoire pour une variable de ce type. Et ce sont justement ces variables qui vont nous être très utiles pour le problème qui nous intéresse ici. Elles ont en effet le format que GSX attend dans les tables de paramètres. Notez par ailleurs que les variables entières sont désignées par le suffixe % ou bien par une déclaration avec l'instruction DEFINT.

Nous vous conseillons de définir les variables entières, dans tout programme BASIC GSX, de la façon suivante :

```
DIM CONTROL%(5),SETPAR%(79),SETPOS%(1,74),GETPAR%(44),GETPOS%(1,74)
```

Veuillez noter, en ce qui concerne la définition des variables, que l'instruction BASIC OPTION BASE 1 ne doit pas apparaître dans un programme sous GSX. OPTION BASE 1 manipule en effet le compteur d'indice interne des variables de façon à ce que la valeur la plus basse ne soit plus zéro mais un.

Mais comment allons-nous pouvoir transmettre maintenant les tables de paramètres à GSX? Le BASIC Mallard met à notre disposition deux instructions différentes pour cela : les instructions USR et CALL. L'instruction USR apparaît inadaptée après une étude plus approfondie. Elle permet en effet d'appeler sans problème des programmes machine mais la transmission de paramètre se limite à un élément de données. Avec l'instruction CALL, il en va tout autrement. Elle nous permet de transmettre très facilement plusieurs paramètres qui seront retransmis de la façon suivante aux registres de l'unité centrale Z80.

Avec un nombre de paramètres inférieur ou égal à 3, la transmission de paramètres s'effectue ainsi :

- HL contient l'adresse du paramètre 1
- DE contient l'adresse du paramètre 2
- BC contient l'adresse du paramètre 3

Comme on envoie en général plus de trois valeurs à GSX, il serait plus intéressant pour la programmation de toujours supposer qu'au moins 4 paramètres à transmettre ont été définis. Les adresses des 4 ou plus paramètres seront transmises de la façon suivante aux registres :

- HL contient l'adresse du paramètre 1
- DE contient l'adresse du paramètre 2
- BC contient l'adresse d'un bloc de paramètres qui contient les adresses des paramètres trois à n-1 en ordre croissant.

Nous indiquions au chapitre 10.3 que GSX attend les adresses de départ des blocs de paramètres nécessaires dans une table dont l'adresse de départ se trouve elle-même dans le registre DE. Si nous examinons de plus près l'organisation de l'instruction CALL, nous constaterons que les adresses des troisième à n-ième éléments de données sont gérées dans un bloc de données de ce type. C'est toutefois le registre BC et non le registre DE qui indique son adresse de départ.

Pour pouvoir employer efficacement l'instruction CALL, nous devons compléter son bloc de variables en le faisant précéder de deux éléments fictifs.

Un élément fictif est une variable dont le contenu peut être quelconque mais dont la présence même est toutefois indispensable pour une bonne exécution de la fonction. Si vous appelez

```
CALL adresse,fictif,fictif,CONTROL%(5),SETPAR%(79),SETPOS%(1,74),GE
TPAR%(44),GETPOS%(1,74)
```

l'adresse du premier bloc de paramètres sera obligatoirement placée dans le bloc de données adressé par le registre BC. HL et DE contiendront les adresses totalement inutiles de nos éléments fictifs.

L'appel de fonction BDOS &0005 ne doit cependant pas être effectué directement à partir du niveau du BASIC pour les raisons suivantes :

1. L'adresse de la table des adresses de départ de bloc de paramètres ne se trouve pas dans le registre DE.

2. Le mot de commande GDOS &73 ne se trouve pas dans le registre C.

Il faut donc appeler d'abord une petite routine machine qui se chargera de régler ces deux points pour nous. Voici comment peut se présenter une routine de ce type :

```
PUSH BC        ;Copier adresse de départ dans DE
POP  DE        ;            "
LD   C,&73     ;Définir le mot de commande GDOS
JP   &0005     ;Appel de fonction de GDOS
```

Il ne nous reste plus maintenant qu'à implanter notre routine machine dans le BASIC Mallard. A cet effet vous devez réserver sept octets de la mémoire de votre AMSTRAD. Il pourrait cependant facilement arriver que le BASIC efface notre routine machine. La réservation du nombre d'octets nécessaire peut être réalisée très simplement avec les deux instructions BASIC MEMORY et HIMEM. HIMEM détermine la dernière adresse utilisée par la mémoire programme du BASIC dans la zone supérieure de la RAM. MEMORY se charge au contraire de fixer l'adresse la plus élevée utilisée par le BASIC.

Comme le BASIC autorise une combinaison pratiquement illimitée de presque toutes les instructions, la séquence d'instructions

```
MEMORY HIMEM-7
```

réservera les sept octets nécessaires dans la mémoire BASIC. HIMEM recevra de ce fait une nouvelle valeur. Nous pouvons maintenant POKEr sans risque notre routine machine dans la mémoire.

```
10 '********************
20 '**                **
30 '** GSX-INIT       **
40 '**                **
50 '********************
60 '
70 MEMORY HIMEM-7
80 FOR I%=1 to 7
90 READ GSX%
100 POKE HIMEM+I,GSX%
110 NEXT I%
120 DATA &HC5,&HD1,&H0E,&H73,&HC3,&H05,&H00
130 DIM CONTROL%(5),SETPAR%(79),SETPOS%(1,74),GETPAR%(44),GETPOS%(1
,74)
140 '
150 REM *** Programme principal ***
```

10.4 LE JEU D'INSTRUCTIONS GSX

Cette section a pour but de vous présenter le jeu d'instructions intégral de GSX. Il s'agit de 20 instructions constituées par les fonctions graphiques mais aussi par des routines chargées de tâches de gestion. Nous avons, par souci de clarté, organisé la description de chaque instruction en quatre rubriques :

1. Nom de la routine et brève description de sa fonction
2. Paramètres d'entrée
3. Paramètres de sortie
4. Description de la fonction

GDOS distingue, comme nous l'indiquions au début, les différentes instructions d'après le contenu du registre CONTROL(0). Il doit contenir avant chaque appel de GSX un numéro d'instruction approprié. Ce numéro d'instruction définissant chaque routine est indiqué, dans la description de chaque routine, sous la rubrique Paramètres d'entrée. Notez bien que ne peuvent être transmises, aussi bien aux registres d'entrée qu'aux registres de sortie, que des valeurs entières.

LOADDR
Charger un nouveau processeur de périphérique

Paramètres d'entrée :

CONTROL(0)	1
CONTROL(1)	0
CONTROL(3)	10

SETPAR(0)	numéro logique du processeur de périphérique à charger
SETPAR(1)	forme de ligne : 1 = continue 2 = petits tirets 3 = pointillée 4 = trait-point 5 = longs tirets 6 = trait-point-point (sauf DDSCREEN)
SETPAR(2)	couleur de ligne : 0 ou 1
SETPAR(3)	forme du marqueur : 1 = . 2 = + 3 = * 4 = O 5 = x
SETPAR(5)	n'est pas utilisé avec LOADDR
SETPAR(6)	couleur de texte : 0 ou 1
SETPAR(7)	mode de remplissage : 0 = squelettique (ne dessiner que le cadre) 1 = tout remplir 2 = avec le modèle de remplissage 3 = avec des hachures
SETPAR(8)	index de hachures : 1 à 6 (voir la table de spécifications de processeur)

SETPAR(9) Couleur de remplissage :
0 ou 1

Paramètres de sortie :

CONTROL(2) contient après appel de la routine GSX le nombre de paramètres transmis à l'utilisateur dans GETPAR.

CONTROL(4) contient après appel d'une routine GSX le nombre de points dont les coordonnées sont transmises dans GETPOS.

GETPAR(0) contient la résolution horizontale dans l'intervalle de 0 à 32767.

GETPAR(1) contient la résolution verticale dans l'intervalle de 0 à 32767.

GETPAR(2) non défini.

GETPAR(3) contient l'extension horizontale d'un point d'image en micromètres.

GETPAR(4) contient l'extension verticale d'un point d'image en micromètres.

GETPOS(1,0) plus petite taille de caractère pouvant être représentée sur le périphérique de sortie.

GETPOS(0,2) plus petite épaisseur de ligne pouvant être représentée sur le périphérique de sortie.

GETPOS(1,4) plus petite taille de marqueur pouvant être représentée sur le périphérique de sortie.

Fonction : LOADDR charge un nouveau processeur de périphérique, qui doit avoir été déclaré dans ASSIGN.SYS, dans la zone de la mémoire réservée à cet effet. L'ancien processeur de périphérique est effacé par cette opération.

Instructions apparentées : Closedr

CLOSEDR
Fermer le processeur de périphérique activé

Paramètres d'entrée :

CONTROL(0) 2
CONTROL(1) 0
CONTROL(3) non défini

Paramètres de sortie :

aucun

Fonction : CLOSEDR met fin au travail avec un processeur sans détruire les paramètres fixés par d'autres instructions graphiques.

Instructions apparentées : Loaddr

CLG
Effacer un dessin

Paramètres d'entrée :

CONTROL(0) 3
CONTROL(1) 0
CONTROL(3) non défini

Paramètres de sortie :

aucun

Fonction : Suivant le processeur de périphérique utilisé, CLG efface l'écran ou bien envoie à l'imprimante un form feed (saut de page). Dans tous les cas, l'effet de cette instruction est de mettre à disposition de l'utilisateur une page blanche.

Instructions apparentées : aucune

OUTPUT

Lire le buffer graphique

Paramètres d'entrée :

CONTROL(0) 4
CONTROL(1) 0
CONTROL(3) non défini

Paramètres de sortie :

aucun

Fonction : Alors que le processeur d'écran DDSCREEN visualise directement toutes les opérations graphiques, toutes les informations graphiques devant être envoyées sur l'imprimante doivent être stockées provisoirement avant la sortie. L'instruction OUTPUT donne l'ordre à GSX de lire cette mémoire provisoire (buffer) et de sortir le résultat sur l'imprimante. Si le processeur d'écran DDSCREEN est activé, cette instruction sera ignorée.

Instructions apparentées : aucune

DRAW
Dessin de ligne(s)

Paramètres d'entrée :

CONTROL(0)	6
CONTROL(1)	nombre de sommets d'angle définis dans SETPOS
CONTROL(3)	non défini
SETPOS(0,0...74)	X1 à Xn
SETPOS(1,0...74)	Y1 à Yn

Paramètres de sortie :

aucun

Fonction : DRAW réunit par une ligne les points définis dans SETPOS. Cette routine tracera, selon la définition des paramètres,

- un point (CONTROL(1)=1),
- une ligne (CONTROL(1)=2)
- ou un polygone (CONTROL(1)>2).

Instructions apparentées : Linestyle, Linecolor

SETMKR
Fixer le marqueur

Paramètres d'entrée :

CONTROL(0)	7
CONTROL(1)	nombre de points définis dans SETPOS
CONTROL(3)	non défini
SETPOS(0,0...74)	X1 à Xn
SETPOS(1,0...74)	Y1 à Yn

Paramètres de sortie :

aucun

Fonction : SETMKR envoie sur l'écran ou dans le buffer d'entrée de l'imprimante un ou plusieurs marqueurs (symboles) dont les coordonnées doivent être définies dans SETPOS.

Instructions apparentées : Mkrstyle, Mkrsize, Mkrcolor

TEXT
Envoyer du texte au périphérique de sortie

Paramètres d'entrée :

CONTROL(0)	8
CONTROL(1)	1
CONTROL(3)	nombre de caractères
SETPAR(0...79)	Texte
SETPOS(0,0...74)	X1 à Xn
SETPOS(1,0...74)	Y1 à Yn

Paramètres de sortie :

aucun

Fonction : TEXT envoie les caractères définis dans SETPAR au périphérique de sortie indiqué. La position du premier caractère doit être fixée avec SETPOS. L'instruction TEXT permet de sortir les 256 caractères des ordinateurs AMSTRAD. Les caractères 0 à 31 ne seront pas ici interprétés comme des codes de contrôle mais comme des caractères spéciaux. Chaque processeur de périphérique dispose d'un mode d'écriture particulier qui diffère selon qu'il s'agit d'une sortie sur écran ou sur imprimante. Sur l'écran, le fond est effacé par la matrice de caractère tout entière (mode AND). L'imprimante ou le plotter ne peuvent par contre, pour des raisons d'ordre technique, sortir sur le papier que les points mis d'un caractère (mode OR). Ces modes ne peuvent être modifiés par l'utilisateur.

Instructions apparentées : Txtsize, Txtdir, Txtcolor

FILLPOLY
Dessiner un polygone plein

Paramètres d'entrée :

CONTROL(0)	9
CONTROL(1)	nombre de sommets d'angle définis dans SETPOS
CONTROL(3)	non défini
SETPOS(0,0...74)	X1 à Xn
SETPOS(1,0...74)	Y1 à Yn

Paramètres de sortie :

aucun

Fonction : FILLPOLY réunit par une ligne, de la même façon que DRAW, les sommets d'angle définis dans SETPOS. Contrairement à ce qui est le cas avec la fonction DRAW, le premier point est également relié par une ligne au dernier point et la surface ainsi délimitée est remplie. GSX ne permet malheureusement pas de remplir n'importe quelle surface. Mais comme vous pouvez définir jusqu'à 75 sommets d'angle, même la représentation d'un cercle plein ne pose pas de problème.

Instructions apparentées : Bar, Fillstyle, Fillindex, Fillcolor

BAR
Dessiner un rectangle

Paramètres d'entrée :

CONTROL(0)	11
CONTROL(1)	2
CONTROL(3)	non défini

SETPOS(0,0)	X1coordonnée du coin inférieur gauche
SETPOS(1,0)	Y1coordonnée du coin inférieur gauche
SETPOS(0,1)	X2coordonnée du coin supérieur droit
SETPOS(1,1)	Y2coordonnée du coin supérieur droit

Paramètres de sortie :

aucun

Fonction : La fonction BAR permet de construire très facilement un rectangle puisqu'il suffit de fournir deux coordonnées de coin pour sa définition. Les rectangles ainsi produits peuvent être remplis de manières très diverses (voir également à ce sujet FILLCOLOR, FILLINDEX, FILLSTYLE).

Instructions apparentées : Fillpoly, Fillstyle, Fillindex, Fillcolor

TXTSIZE
Déterminer la taille de caractère

Paramètres d'entrée :

CONTROL(0)	12
CONTROL(1)	1
CONTROL(3)	non défini
SETPOS(0,0)	0
SETPOS(1,0)	Longueur du texte (1 à 12)

Paramètres de sortie :

aucun

Fonction : TXTSIZE permet de déterminer la taille de caractère du texte à sortir. Si le processeur d'écran DDSCREEN est activé, cette instruction sera ignorée car il ne peut traiter que la taille de caractère "1".

Instructions apparentées : Text, Txtdir, Txtcolor, Mkrsize

TXTDIR
Fixer la direction du texte

Paramètres d'entrée :

CONTROL(0)	13
CONTROL(1)	0
CONTROL(3)	non défini
SETPAR(0)	1 = 0 degré
	2 = 270 degrés
	3 = 180 degrés
	4 = 90 degrés

Paramètres de sortie :

aucun

Fonction : TXTDIR fixe la direction de sortie d'un texte. Vous disposez à cet effet de quatre directions étagées par intervalles de 90 degrés chaque fois. Ces directions doivent être indiquées à l'aide de numéros de fonction correspondant aux différents angles de rotation. TXTDIR n'est pas implantée dans le processeur écran DDSCREEN et sera donc ignorée si ce processeur est activé.

Instructions apparentées : Text, Txtsize, Txtcolor

LINESTYLE
Fixer la forme de ligne

Paramètres d'entrée :

CONTROL(0)	15
CONTROL(1)	0
CONTROL(3)	non défini
SETPAR(0)	1 = continue
	2 = petits tirets
	3 = pointillée
	4 = trait-point
	5 = longs tirets
	6 = trait-point-point

Paramètres de sortie :

aucun

Fonction : LINESTYLE fixe l'apparence d'une ligne. Cette instruction permet de définir très simplement des lignes de symétrie, de démarcation ou autre. La forme de la ligne est fixée à cet effet à l'aide de SETPAR(0). Il faut absolument tenir compte, pour le choix de forme de ligne, du fait que le processeur d'écran DDSCREEN ne peut pas représenter de ligne trait-point-point.

Instructions apparentées : Draw, Linecolor

LINECOLOR
Définir une couleur de ligne

Paramètres d'entrée :

CONTROL(0) 17
CONTROL(1) 0
CONTROL(3) non défini

SETPAR(0) 0 = annuler point
1 = fixer point

Paramètres de sortie :

aucun

Fonction : LINECOLOR fixe si une ligne doit être dessinée ou effacée. Si un 1 figure dans SETPAR(0) lors de l'appel de GSX, toutes les lignes suivantes seront désormais représentées en vert sur l'écran et sur l'imprimante dans la couleur du ruban de couleur utilisé. Si l'utilisateur transmet par contre un 0, le dessin se fera désormais dans la couleur du fond. Cela se traduira sur l'écran mais n'aura aucun effet sur l'imprimante ou le plotter.

Instructions apparentées : Draw, Linestyle

MKRSTYLE
Fixer la forme du marqueur

Paramètres d'entrée :

CONTROL(0)	18
CONTROL(1)	0
CONTROL(3)	non défini

SETPAR(0)
1 = .
2 = +
3 = *
4 = O
5 = x

Paramètres de sortie :

aucun

Fonction : MKRSTYLE permet de fixer la forme d'un symbole (le marqueur) qui pourra être utilisé comme marque dans les graphiques ou autres dessins. Le code correspondant à la forme voulue doit être transmis dans SETPAR(0).

Instructions apparentées : Setmkr, Mkrsize, Mkrcolor

MKRSIZE
Fixer la taille du marqueur

Paramètres d'entrée :

CONTROL(0)	19
CONTROL(1)	1
CONTROL(3)	non défini
SETPOS(0,0)	0
SETPOS(1,0)	taille du marqueur (1 à 12)

Paramètres de sortie :

aucun

Fonction : MKRSIZE permet de fixer la taille de caractère des marqueurs (symboles). Si le processeur d'écran DDSCREEN est activé, cette instruction sera ignorée car ce processeur ne peut traiter que la taille de caractère "1".

Instructions apparentées : Setmkr, Mkrstyle, Mkrcolor, Txtsize

MKRCOLOR
Définir la couleur du marqueur

Paramètres d'entrée :

CONTROL(0) 20
CONTROL(1) 0
CONTROL(3) non défini

SETPAR(0) 0 = annuler point
1 = fixer point

Paramètres de sortie :

aucun

Fonction : MKRCOLOR fixe si un marqueur doit être dessiné ou effacé. Si un 1 figure dans SETPAR(0) lors de l'appel de GSX, tous les marqueurs suivants seront désormais représentés en vert sur l'écran et sur l'imprimante dans la couleur du ruban de couleur utilisé. Si l'utilisateur transmet par contre un 0, le dessin se fera désormais dans la couleur du fond. Cela se traduira sur l'écran mais n'aura aucun effet sur l'imprimante ou le plotter.

Instructions apparentées : Setmkr, Mkrstyle, Mkrsize

TXTCOLOR
Définir la couleur de texte

Paramètres d'entrée :

CONTROL(0)	22
CONTROL(1)	0
CONTROL(3)	non défini
SETPAR(0)	0 = annuler point 1 = fixer point

Paramètres de sortie :

aucun

Fonction : TXTCOLOR fixe si un ou plusieurs caractères doivent être dessinés ou effacés. Si un 1 figure dans SETPAR(0) lors de l'appel de GSX, tous les caractères suivants seront désormais représentés en vert sur l'écran et sur l'imprimante dans la couleur du ruban de couleur utilisé. Si l'utilisateur transmet par contre un 0, la sortie se fera désormais dans la couleur du fond. Cela se traduira sur l'écran mais n'aura aucun effet sur l'imprimante ou le plotter.

Instructions apparentées : Text, Txtsize, Txtdir

FILLSTYLE
Fixer le mode de remplissage

Paramètres d'entrée :

CONTROL(0)	23
CONTROL(1)	0
CONTROL(3)	non défini
SETPAR(0)	0 = squelettique (ne dessiner que le cadre)
	1 = tout remplir
	2 = avec le modèle de remplissage
	3 = avec des hachures

Paramètres de sortie :

aucun

Fonction : FILLSTYLE permet de déterminer de quelle manière des surfaces closes doivent être remplies. Le code correspondant au mode de remplissage doit être transmis dans SETPAR(0).

Instructions apparentées : Fillpoly, Bar, Fillindex, Fillcolor

FILLINDEX
Définir le modèle de remplissage ou les hachures

Paramètres d'entrée :

CONTROL(0) 24
CONTROL(1) 0
CONTROL(3) non défini

SETPAR(0) Index de hachures ou de modèle: 1 à 6 (voir la table de spécification du processeur)

Paramètres de sortie :

aucun

Fonction : La routine GSX FILLINDEX permet de définir des hachures ou des modèles de remplissage qui seront utilisés par les fonctions de remplissage automatique de surfaces. L'index de remplissage, qui doit être transmis dans SETPAR(0), correspond soit à un des 6 modèles de remplissage existants soit aux hachures. La fonction FILLSTYLE doit avoir été utilisée auparavant pour spécifier si l'index de remplissage se rapporte aux hachures ou s'il doit être ignoré. Vous trouverez les différents modèles dans la table de spécification de processeur que nous vous présenterons plus loin. FILLINDEX n'a malheureusement pas d'effet sur le processeur d'écran DDSCREEN. Cette instruction sera donc ignorée pour la sortie écran et la surface à remplir sera simplement remplie avec la couleur du fond.

Instructions apparentées : Fillpoly, Bar, Fillstyle, Fillcolor

FILLCOLOR
Définir la couleur de surface

Paramètres d'entrée :

CONTROL(0) 25
CONTROL(1) 0
CONTROL(3) non défini

SETPAR(0) 0 = annuler point
1 = fixer point

Paramètres de sortie :

aucun

Fonction : FILLCOLOR fixe si une surface doit être dessinée ou effacée. Si un 1 figure dans SETPAR(0) lors de l'appel de GSX, toutes les surfaces suivantes seront désormais représentées en vert sur l'écran et sur l'imprimante dans la couleur du ruban de couleur utilisé. Si l'utilisateur transmet par contre un 0, le remplissage se fera désormais dans la couleur du fond. Cela se traduira sur l'écran mais n'aura aucun effet sur l'imprimante ou le plotter.

Instructions apparentées : Fillpoly, Bar, Fillstyle, Fillindex

10.5 GDOS DE L'INTERIEUR

Ce chapitre a pour but de familiariser le programmeur en langage machine avec la structure interne de GDOS. Nous allons à cet effet vous présenter ici un commentaire complet du listing de GDOS. La zone mémoire utilisée pour établir ce listing (adresse de départ BE00 hexa) se rapporte à la configuration de processeur standard définie par DIGITAL RESEARCH :

- DDFXHR8
- DDFXLR8
- DDHP7070
- DDSCREEN

Les processeurs ont été installés avec le programme BASIC.COM. Si vous préférez utiliser une autre configuration logicielle dans vos programmes, vous aurez très probablement à rechercher l'adresse de départ de GSX à l'aide de programmes de recherche par tâtonnement.

Si vous jetez un rapide coup d'oeil sur notre commentaire du listing, vous remarquerez immédiatement que les mnémoniques Z80 y font défaut. Comme c'était déjà le cas pour le listing de la ROM présenté au chapitre 9.6, la législation sur la propriété littéraire et artistique ne nous permet pas d'imprimer ici les codes d'opération ni les mnémoniques. Pour que vous puissiez malgré tout consulter le listing GDOS complet (adresses, codes d'opération, mnémoniques et commentaires), nous vous présentons ici un désassembleur pour le PCW. Les possesseurs d'un 6128 pourront réutiliser le désassembleur présenté au chapitre 9.6.1. Il leur suffira dans ce cas de remplacer la lecture de la mémoire ROM par une simple fonction PEEK.

Le désassembleur présenté ici traduit le code machine en mnémoniques Z80. Nous avons totalement renoncé à présenter ici les mnémoniques 8080 car le code symbolique de ce processeur est bien moins compréhensible. Votre PCW n'est d'ailleurs pas doté d'un microprocesseur 8080 mais bien d'un Z80.

Une fois le désassemblage effectué, les spécialistes du langage machine s'étonneront peut-être du style de programmation très laborieux.

Ce code très inefficace n'est pas dû à l'incapacité des programmeurs mais bien plutôt au fait que GDOS a été entièrement programmé en langage machine 8080. Or ce microprocesseur ne dispose que d'un dixième des instructions du jeu d'instructions du Z80.

```
50 rem désassembleur pour le PCW
60 rem
70 GOTO 1020
80 PRINT"D E S A S S E M B L E U R - Z 8 0"
90 PRINT:PRINT:INPUT"Adresse de départ : &h",a$
100 GOSUB 930:debut=a
110 PRINT:INPUT"Adresse finale : &h",a$
120 GOSUB 930:fin=a
130 IF debut>fin THEN 60
140 pc=debut
150 adr=pc
160 PRINT HEX$(adr,4);"  ";
170 iflag=0
180 GOSUB 970
190 GOSUB 310
200 IF iflag THEN 620
210 IF (w=&HCF OR w=&HD7 OR w=&HDF OR w=&HEF) AND (LEFT$(pr$,3)="RS
T") THEN pr$=pr$+" /DW:nn"
220 IF INSTR(pr$,"n")<>0 THEN 730
230 IF INSTR(pr$,"e")<>0 THEN 850
240 po=INSTR(pr$," ")
250 IF PR$="" THEN PR$="???"
260 IF po=0 THEN PRINT TAB(21);pr$;:GOTO 280
270 PRINT TAB(21);LEFT$(pr$,po-1);TAB(27);RIGHT$(pr$,LEN(pr$)-po);
280 PRINT
290 IF pc<=fin THEN 150
300 END
310 REM Interpréter
320 IF (w=&HDD OR w=&HFD) AND NOT iflag THEN 510
330 IF w=&HED THEN 480
340 IF w=&HCB THEN 420
350 GOSUB 560
360 ON co1 GOTO 380,400,370
370 pr$=bef$(w):RETURN
380 IF w=&H76 THEN pr$="HALT":RETURN
390 pr$="LD "+regtab$(co2)+","+reg$:RETURN
```

```
400 IF co2=0 OR co2=1 OR co2=3 THEN a$=" A," ELSE a$=" "
410 pr$=arilog$(co2)+a$+reg$:RETURN
420 REM cb
430 GOSUB 970
440 IF iflag THEN dis=w:GOSUB 970
450 GOSUB 560
460 IF co1=0 THEN pr$=rotschi$(co2)+" "+reg$ ELSE pr$=bitti$(co1)+S
TR$(co2)+","+reg$
470 RETURN
480 REM ed
490 GOSUB 970
500 IF w<&H40 OR w>&HBF THEN pr$="???":RETURN ELSE GOTO 370
510 REM xy
520 iflag=-1
530 IF w=&HDD THEN i$="IX" ELSE i$="IY"
540 GOSUB 970
550 GOTO 310
560 REM décomposer le code
570 co1=(w AND 192)/64
580 co2=(w AND 56)/8
590 co3=w AND 7
600 reg$=regtab$(co3)
610 RETURN
620 REM indexé
630 po=INSTR(pr$,"HL")
640 IF po=0 THEN pr$="???":GOTO 240
650 IF INSTR(pr$,"(HL)")<>0 THEN 690
660 IF pr$="EX DE,HL" THEN pr$="???":GOTO 240
670 IF pr$="ADD HL,HL" THEN pr$="ADD "+i$+","+i$:GOTO 240
680 pr$=LEFT$(pr$,po-1)+i$+RIGHT$(pr$,LEN(pr$)-po-1):GOTO 210
690 IF LEFT$(pr$,2)="JP" THEN 680
700 IF pc-adr<3 THEN GOSUB 970:dis=w
710 IF dis>127 THEN dis$=STR$(dis-
256) ELSE dis$="+"+RIGHT$(STR$(dis),LE N(STR$(dis))-1)
720 i$=i$+dis$:GOTO 680
730 REM remplacer n
740 po=INSTR(pr$,"nn")
750 IF po<>0 THEN 800
760 po=INSTR(pr$,"n")
770 GOSUB 970
780 pr$=LEFT$(pr$,po-1)+"&"+HEX$(w,2)+RIGHT$(pr$,LEN(pr$)-po)
790 GOTO 240
```

```
800 GOSUB 970:lb=w
810 GOSUB 970
820 wert=w*256+lb
830 pr$=LEFT$(pr$,po-1)+"&"+HEX$(wert,4)+RIGHT$(pr$,LEN(pr$)-po-1)
840 GOTO 240
850 REM remplacer e
860 po=INSTR(pr$,"e")
870 GOSUB 970
880 IF w>127 THEN w=w-256:REM complément à 2
890 w=w+2
900 a$="$"+STR$(w)+" >"+"&"+HEX$(pc+w-2,4)
910 pr$=LEFT$(pr$,po-1)+a$+RIGHT$(pr$,LEN(pr$)-po)
920 GOTO 240
930 REM conversion hexa -> déc
940 IF a$="" THEN a=0:RETURN
950 a=VAL("&h"+a$)
960 RETURN
970 REM lire octet
980 w=PEEK(pc)
990 pc=pc+1
1000 PRINT HEX$(w,2);" ";
1010 RETURN
1020 REM init
1030 DIM regtab$(12),rotschi$(8),bitti$(3),arilog$(7),bef$(255)
1040 FOR i=0 TO 7:READ regtab$(i):NEXT
1050 FOR i=0 TO 7:READ rotschi$(i):NEXT
1060 FOR i=1 TO 3:READ bitti$(i):NEXT
1070 FOR i=0 TO 7:READ arilog$(i):NEXT
1080 FOR i=0 TO &H7F:READ bef$(i):NEXT
1090 FOR i=&H80 TO &H9F:bef$(i)="":NEXT
1100 FOR i=&HA0 TO &HFF:READ bef$(i):NEXT
1110 GOTO 80
1120 REM DATAS
1130 DATA B,C,D,E,H,L,(HL),A
1140 DATA RLC,RRC,RL,RR,SLA,SRA,???,SRL
1150 DATA BIT,RES,SET
1160 DATA ADD,ADC,SUB,SBC,AND,XOR,OR,CP
1170 DATA NOP,"LD BC,nn","LD (BC),A",INC BC,INC B,DEC B,"LD
B,n",RLCA
1180 DATA "EX AF,AF'","ADD HL,BC","LD A,(BC)",DEC BC,INC C,DEC C,"L
D C,n",RRCA
1190 DATA DJNZ e,"LD DE,nn","LD (DE),A",INC DE,INC D,DEC D,"LD
```

```
D,n",RLA
1200 DATA JR e,"ADD HL,DE","LD A,(DE)",DEC DE,INC E,DEC E,"LD
E,n",RRA
1210 DATA "JR NZ,e","LD HL,nn","LD (nn),HL",INC HL,INC H,DEC H,"LD
H,n",DAA
1220 DATA "JR Z,e","ADD HL,HL","LD HL,(nn)",DEC HL,INC L,DEC L,"LD
L,n",CPL
1230 DATA "JR NC,e","LD SP,nn","LD (nn),A",INC SP,INC (HL),DEC (HL)
,"LD (HL),n",SCF
1240 DATA "JR C,e","ADD HL,SP","LD A,(nn)",DEC SP,INC A,DEC A,"LD A
,n",CCF
1250 DATA "IN B,(C)","OUT (C),B","SBC HL,BC","LD (nn),BC",NEG,RETN,
IM 0,"LD I,A"
1260 DATA "IN C,(C)","OUT (C),C","ADC HL,BC","LD BC,(nn)",,RETI,,"L
D R,A"
1270 DATA "IN D,(C)","OUT (C),D","SBC HL,DE","LD (nn),DE",,,IM 1,"L
D A,I"
1280 DATA "IN E,(C)","OUT (C),E","ADC HL,DE","LD DE,(nn)",,,IM 2,"L
D A,R"
1290 DATA "IN H,(C)","OUT (C),H","SBC HL,HL","LD (nn),HL",,,,RRD
1300 DATA "IN L,(C)","OUT (C),L","ADC HL,HL","LD HL,(nn)",,,,RLD
1310 DATA ,,"SBC HL,SP","LD (nn),SP",,,,
1320 DATA "IN A,(C)","OUT (C),A","ADC HL,SP","LD SP,(nn)",,,,
1330 DATA LDI,CPI,INI,OUTI,,,,,LDD,CPD,IND,OUTD,,,,
1340 DATA LDIR,CPIR,INIR,OTIR,,,,,LDDR,CPDR,INDR,OTDR,,,,
1350 DATA RET NZ,POP BC,"JP NZ,nn",JP nn,"CALL NZ,nn",PUSH BC,"ADD
A,n",RST &00
1360 DATA RET Z,RET,"JP Z,nn",->,"CALL Z,nn",CALL nn,"ADC A,n",RST
&08
1370 DATA RET NC,POP DE,"JP NC,nn","OUT (n),A","CALL NC,nn",PUSH DE
,"SUB n",RST &10
1380 DATA RET C,EXX,"JP C,nn","IN A,(n)","CALL C,nn",-
>,"SBC A,n",RST &18
1390 DATA RET PO,POP HL,"JP PO,nn","EX (SP),HL","CALL PO,nn",PUSH H
L,"AND n",RST &20
1400 DATA RET PE,JP (HL),"JP PE,nn","EX DE,HL","CALL PE,nn",-
>,"XOR n",RST &28
1410 DATA RET P,POP AF,"JP P,nn",DI,"CALL P,nn",PUSH AF,"OR n",RST
&30
1420 DATA RET M,"LD SP,HL","JP M,nn",EI,"CALL M,nn",->,"CP n",RST
&38
```

GDOS 1.0

BE00 ENTREE GDOS

BE00 ;Entrée GDOS
BE03 ;Entrée BDOS
;
BE06 ;Numéro de périphérique du processeur activé

BE08 PROCESSEUR DE PERIPHERIQUE 1

BE08 ;Numéro de périphérique (21)
BE0A ;Lecteur
BE0B ;Désignation de périphérique
BE12 ;Espace pour 8ème position

BE13 PROCESSEUR DE PERIPHERIQUE 2

BE13 ;Numéro de périphérique (22)
BE15 ;Lecteur
BE16 ;Désignation de périphérique
BE1D ;Espace pour 8ème position

BE1E PROCESSEUR DE PERIPHERIQUE 3

BE1E ;Numéro de périphérique (11)
BE20 ;Lecteur
BE21 ;Désignation de périphérique

BE29 PROCESSEUR DE PERIPHERIQUE 4

BE29 ;Numéro de périphérique (1)
BE2B ;Lecteur
BE2C ;Désignation de périphérique

```
BE34                      PROCESSEUR DE PERIPHERIQUE 5

BE34    ;Marque de fin car pas de cinquième processeur déclaré
BE36    ;Lecteur
BE37    ;Place réservée
BE39    ;pour désignation de périphérique
BE3B    ;         "
BE3D    ;         "
        ;
BE3F    ;Marque de fin


BE41                      MESSAGE GSX

BE41    "------------------------------"
BE5F    "---------------------"
        ;
BE74    ;Marque de fin de ligne
        ;
BE76    "GSX-80 1.1  01 Oct 83    Seria"
BE94    "l No 5000-1232-654321"
        ;
BEA9    ;Marque de fin de ligne
        ;
BEAB    "Copyright (C) 1983            "
BEC9    "                    "
        ;
BEDE    ;Marque de fin de ligne
        ;
BEE0    "Digital Research, Inc.        "
BEFE    "    All rights reserved"
        ;
BF13    ;Marque de fin de ligne
        ;
BF15    "------------------------------"
BF33    "---------------------
        ;
BF48    ;Marque de fin de ligne
        ;
BF4A    ;Octet EM
        ;
BF4B    ;Buffer
```

BF6D DEBUT DE GDOS

BF6D ;S'agit-il d'un
BF6E ;appel de GSX?
BF70 ;Si non, sauter au BDOS
;
BF73 ;HL = début du buffer
BF76 ;Aller chercher les adresses de départ des 5 vecteurs d'entrée/sortie GSX (cinq adresses = 10 octets)
BF78 ;Appeler TRANSBYT
;
BF7B ;HL = Adresse de CONTROL(0)
BF7E ;A = Octet faible de CONTROL(0)
BF7F ;Pointer sur octet fort
BF80 ;H = Octet fort (CONTROL(0))
BF81 ;HL = CONTROL(0)
BF82 ;Numéro de fonction -1 pour comparaison
BF83 ;Charger nouveau
BF84 ;processeur de périphérique?
BF85 ;Si non continuer
;
BF88 ;Appeler GETCNT
BF8B ;Appeler processeur de périphérique
BF8E ;Appeler SENDPAR

BF91 CHARGER NOUVEAU PROCESSEUR DE PERIPHERIQUE

BF91 ;HL = Adresse de SETPAR(0)
BF94 ;E = Octet faible de SETPAR (0)
BF95 ;Pointer sur Octet fort
BF96 ;DE = nouveau numéro de périphérique
BF97 ;Aller chercher actuel numéro de périphérique
BF9A ;Appeler CMPDRV
BF9D ;Si nouveau processeur = processeur actuel, sortir erreur!
BFA0 ;Pointeur sur numéro de périphérique de processeur 1
BFA3 ;Sauver adresse
BFA4 ;Aller chercher octet faible du numéro de périphérique
BFA5 ;Aller chercher octet fort du
BFA6 ;numéro de périphérique
BFA7 ;HL = numéro de périphérique
BFA8 ;numéro de périphérique

BFA9	;valable?
BFAA	;Remettre la pile en ordre
BFAB	;Sortir erreur?
BFAE	;Sauver pointeur
BFAF	;Appeler CMPDRV
BFB2	;Pointeur sur numéro de périphérique
BFB3	;Si processeur de périphérique déclaré sauter en &BFBD
BFB6	;Offset pour prochain numéro de périphérique
BFB9	;Calculer adresse du prochain numéro de périphérique
BFBA	;aller chercher prochain numéro
	;
BFBD	;aller chercher numéro de lecteur
BFBE	; "
BFBF	;aller chercher nouveau numéro de périphérique
BFC0	;numéro de périphérique actuel= nouveau numéro de périphérique!
BFC3	;Buffer pour READFILE
BFC6	;Transférer 9 octets (numéro de lecteur et nom de processeur de périphérique)
BFC8	;Appeler TRANSBYTE
BFCB	;Appeler READFILE (charger processeur)
BFCE	;Appeler ERROR erreur apparue?
BFD1	;HL = Adresse de GETPAR(0)
BFD4	;Aller chercher octet faible
BFD5	;Pointer sur octet fort
BFD6	;et sauver adresse
BFD7	;H = Octet fort
BFD8	;HL = GETPAR(0)+1
BFD9	; "
BFDA	;et ranger
BFDD	;Aller chercher octet fort de l'adresse de GETPAR(0)
BFDE	;Pointer sur octet faible de l'adresse GETPAR(1)
BFDF	;Aller chercher octet faible
BFE0	;Pointer sur octet fort
BFE1	;H = Octet fort
BFE2	;HL = GETPAR(1)+1
BFE3	; "
BFE4	;et ranger
BFE7	;Saut à Jump

BFEA		GETCNT

```
BFEA    ;HL = Adresse de CONTROL(0)
BFED    ;Pointeur sur adresse
BFEE    ;de CONTROL(1)
BFEF    ;E = Octet faible de CONTROL(1)
BFF0    ;Pointer sur octet fort
BFF1    ;DE = CONTROL(1)
BFF2    ;Des coordonnées sont-elles
BFF3    ;transmises?
BFF4    ;Si non, retour au programme principal
BFF5    ;Offset pour comparaison
BFF8    ;Plus de 75 coordonnées
BFF9    ;ont-elles été transmises?
BFFC    ;Si oui, ramener leur nombre à 75
BFFE    ;Pointeur sur adresse de SETPOS(0)
C001    ;fixer et sauver
C002    ;Organiser buffer GET
C005    ;et ranger adresse de départ
C008    ;Transférer pointeur
C009    ;dans BC
C00A    ;Aller chercher adresse de SETPOS(0)
        ;
C00B    ;Sauver le nombre de coordonnées
C00C    ;DE = Adresse de SETPOS(0)
C00D    ;Aller chercher GETPAR(0)
C010    ;Appeler PARCONV convertir GETPAR(n)
C013    ;DE = Adresse de SETPOS(1)
C014    ;Aller chercher GETPAR(1)
C017    ;Appeler PARCONV
C01A    ;Aller chercher nombre de coordonnées
C01B    ;Nombre = nombre - 1
C01C    ;Y a-t-il encore des coordonnées?
        ;
C01F    ;Terminé!
```

C020		PARCONV

```
C020    ;Envoyer adresse du SETPOS dans HL
C021    ;Accu = Octet faible de SETPOS(0)
C022    ;Pointeur sur octet fort
```

```
C023    ;et sauver
C024    ;H = Octet fort de SETPOS(0)
C025    ;sauver pointeur de buffer GET
C026    ;HL = SETPOS(0)
C027    ;initialiser compteur
C029    ;sauver GETPAR(0) ou GETPAR(1)
C02A    ;SETPOS(0) dans DE
C02B    ;initialiser HL
        ;
        ;***************** SETPOS(n)/15
        ;
C02E    ;Aller chercher octet fort de SETPOS(n)
C02F    ;le diviser par 2
C030    ;et écrire le résultat dans D
C031    ;Aller chercher octet faible de SETPOS(n) et
C032    ;le diviser par 2
C033    ;DE = SETPOS(n)/2
C034    ;Sauter si aucun reste n'est apparu lors de la division
C037    ;sauver compteur
C038    ;Aller chercher GETPAR(0) ou GETPAR(1)
C039    ;et renvoyer sur la pile
C03A    ;HL <-- BC
C03B    ;Renvoyer le compteur dans C
        ;
C03C    ;Aller chercher octet fort
C03D    ;le diviser par 2
C03E    ;et renvoyer résultat dans H
C03F    ;Aller chercher octet faible
C040    ;Le diviser par 2
C041    ;et renvoyer le résultat dans H
C042    ;Compteur = compteur - 1
C043    ;Fin de la boucle?
        ;
C046    ;Aller chercher buffer GET
C047    ;       "
C048    ;Envoyer octet faible
C049    ;       "
C04A    ;Augmenter pointeur
C04B    ;Aller chercher octet fort
C04C    ;et l'écrire dans le buffer GET
C04D    ;Augmenter pointeur
C04E    ;Aller chercher SETPOS(n)
```

C04F ;Pointer sur n+1
C050 ;Fin de la routine

C051 SENDPAR

C051 ;Aller chercher adresse de CONTROL(0)
C054 ;Pointer sur adresse de CONTROL(4)
C057 ; "
C058 ;Aller chercher octet faible de CONTROL(4)
C059 ;Aller chercher octet fort
C05A ;BC = CONTROL(4), nombre de coordonnées envoyées
;à l'utilisateur
C05B ;Aller chercher adresse de GETPOS(0)
;
C05E ;Faut-il envoyer
C05F ;des coordonnées?
C060 ;Si non, retour au programme principal
C061 ;Sauver CONTROL(4)
C062 ;DE = Adresse de GETPOS(n,n)
C063 ;Aller chercher GETPAR(0)
C066 ;Appeler PARTRANS
C069 ;DE = GETPOS(n+1,n)
C06A ;Aller chercher GETPAR(1)
C06D ;Appeler PARTRANS
C070 ;Aller chercher le nombre de coordonnées à envoyer
C071 ;et diminuer ce nombre de un
C072 ;Traiter les paramètres suivants

C075 PARTRANS

C075 ;BC = GETPAR(0) ou
C076 ;GETPAR(1)
C077 ;Aller chercher adresse de GETPOS(n,n)
C078 ;E = Octet faible de GETPOS(n,n)
C079 ;sauver adresse de GETPOS(n,n)
C07A ;Aller chercher octet fort
C07B ;DE = GETPOS(n)
C07C ;HL = GETPOS(n)
C07D ;Initialiser compteur
C07F ;Initialiser NEW.PRM

C082	;Sauver compteur
C083	;Aller chercher NEW.PRM
C084	;et multiplier par 2
C085	;Aller chercher GETPOS(n)
C086	;GETPOS * 2
C087	;Accu = Octet faible
C088	;moins octet faible de GETPAR(0) ou GETPAR(1)
C089	;et renvoyer
C08A	;Accu = Octet fort
C08B	;moins Carry et octet fort de GETPAR(0) ou GETPAR(1)
C08C	;HL = résultat
C08D	;inférieur à zéro?
C090	;Ajouter au résultat GETPAR(0) ou GETPAR(1)
C091	;Protéger NEW.PRM contre toute modification (INC DE)
C092	;NEW.PRM + 1
C093	;Aller chercher compteur
C094	;et l'augmenter
C095	;Compteur différent de zéro?
C098	;Annuler flag Carry
C099	;et aller chercher NEW.PRM
C09A	;NEW.PRM/2
C09B	; "
C09C	; "
C09D	; "
C09E	; "
C09F	;Débordement?
C0A2	;Si non augmenter
C0A3	;Aller chercher adresse de GETPOS(n,n)
C0A4	;Stocker NEW.PRM
C0A5	; "
C0A6	; "
C0A7	;Pointer sur GETPOS(1,n) ou GETPOS(0,n+1)
C0A8	;Fin de la routine

C0A9 TRANSBYT

TRANSBYT transfère un bloc de données d'une longueur maximum de 256 octets dans un autre zone de la RAM.

Paramètres d'entrée :	*Paramètres de sortie :*
DE = Adresse de départ du bloc	Aucun
HL = Adresse de destination	
C = Longueur de bloc en octets	

Registres modifiés : A, C, D, E, F, H, L

C0A9	;Aller chercher octet...
C0AA	;et le transférer dans le buffer
C0AB	;Pointer sur l'octet suivant
C0AC	;Pointer sur l'octet suivant
C0AD	;Tous les octets ont été transférés?
C0AE	;Si oui retour au programme principal
C0B1	; "

C0B2 CMPDRV

CMPDRV compare le numéro du processeur de périphérique actuellement utilisé avec celui du nouveau processeur. Le numéro de périphérique actuel doit figurer ici dans HL et le nouveau dans DE. Si les deux numéros sont identiques, le flag Zéro sera mis après exécution de la routine, si ce n'est pas le cas, ce flag sera annulé.

Paramètres d'entrée :	*Paramètres de sortie :*
DE = nouveau numéro de périphérique	aucun
HL = actuel numéro de périphérique	

Registres modifiés : A, F, H, L

C0B2	;Accu = Octet faible du numéro de périphérique actuel
C0B3	;soustraire octet faible du nouveau processeur de périphérique
C0B4	;Sauver le résultat
C0B5	;Accu = Octet fort du numéro de périphérique actuel
C0B6	;soustraire octet fort + retenue du nouveau processeur de périphérique
C0B7	;Sauver résultat
C0B8	;Nouveau processeur = processeur actuel?
C0B9	;Terminé!

C0BA BUFFER SYSTEME

C0BA	Adresse de départ CONTROL
C0BB	; "
C0BC	Adresse de départ SETPAR
C0BD	; "
C0BE	Adresse de départ SETPOS
C0BF	; "
C0C0	Adresse de départ GETPAR
C0C1	; "
C0C2	Adresse de départ GETPOS
C0C3	; "
C0C4	Adresse de départ GETPAR(0)
C0C5	; "
C0C6	Adresse de départ GETPAR(1)
C0C7	; "

10.6 MESSAGES D'ERREUR DE GSX

GSX possède tout une palette de messages spéciaux qui sont automatiquement sortis sur l'écran lorsqu'une erreur apparaît. La sortie d'un de ces messages d'erreur peut entraîner une interruption du programme avec retour au système d'exploitation CP/M.

Message d'erreur	Explication
d:nom de fichier.PRL not found	"Processeur de périphérique non trouvé" : le processeur de périphérique déclaré dans ASSIGN.SYS ne figure pas sur le lecteur prédéfini. Le processeur voulu doit être copié sur la disquette du lecteur concerné ou bien la spécification de lecteur dans ASSIGN .SYS doit être modifiée.
d:nom de fichier.PRL empty	"Processeur de périphérique vide" : le processeur de périphérique déclaré dans ASSIGN.SYS a bien été trouvé mais son contenu est erroné ou vide. Le processeur indiqué dans le message d'erreur doit être copié à nouveau sur la disquette concernée.
d:nom de fichier.PRL contains absolute segment	"Processeur de périphérique contient des données incompatibles" : le processeur de périphérique concerné contient des données erronées. Le processeur indiqué dans le message d'erreur doit être copié à nouveau sur la disquette concernée.

d:nom de fichier.PRL closing error

"Processeur de périphérique non refermé" : la disquette portant le processeur de périphérique indiqué dans le message d'erreur a été changée. Elle doit être replacée dans le lecteur de disquette déclaré dans ASSIGN.SYS.

d:nom de fichier.PRL load error

"Erreur de chargement de processeur de périphérique" : les processeurs de périphérique déclarés dans ASSIGN.SYS n'ont pas été inscrits par ordre décroissant de taille. ASSIGN.SYS doit donc être à nouveau défini.

10.7 SPECIFICATIONS DE PROCESSEUR DE PERIPHERIQUE

Chaque processeur de périphérique est doté par son électronique propre de caractéristiques différentes. Les programmeurs de DIGITAL RESEARCH se sont efforcés d'estomper ces différences au maximum mais dans certains domaines il est difficile d'obtenir une large compatibilité. C'est ainsi qu'une imprimante ne peut par exemple pas effacer une ligne. Sur l'écran, il sera, de même, impossible de réaliser un saut de page. Il en résulte que certains programmes ne pourront fonctionner correctement qu'avec des processeurs de périphérique bien précis.

Dans de nombreux cas on pourra cependant fort bien utiliser le processeur d'écran DDSCREEN en liaison avec un processeur d'imprimante. Cette combinaison présente l'avantage que le dessin à traiter sur l'écran peut être saisi et manipulé, dans sa structure de base, sans aucune difficulté. Une fois que toutes les opérations se seront déroulées sans encombre, rien ne s'opposera à une sortie sur votre périphérique.

Pour que vous puissiez vous faire une idée des possibilités des différents processeurs de périphérique, voici maintenant des tables de spécifications de processeur qui vous fourniront toutes les informations nécessaires à ce sujet. Ces tables ne sont d'ailleurs pas intéressantes seulement pour les programmeurs et elles pourront être très utiles au simple utilisateur souhaitant acquérir un logiciel car certains programmes GSX exigent que le processeur de périphérique implanté possède des caractéristiques déterminées.

DDSCREEN.PRL

Périphérique de sortie :	Ecran
Résolution :	720 x 248 points d'image
Forme de ligne :	1 continue 2 petits tirets 3 pointillée 4 trait-point (ligne de symétrie) 5 longs tirets
Epaisseurs de trait :	1
Marqueur (Symboles) :	1 . 2 + 3 * 4 o 5 x
Tailles de caractère :	une (taille standard)
Directions de texte :	une (0)
Modèle de remplissage :	aucun (la surface est encadrée)
Hachures :	aucune (la surface est encadrée)
Elément général de représentation :	contours de barre

Séquences Escape :	toutes sauf tableur graphique et copie d'écran imprimée
Entrée :	non

DDFXLR8.PRL

Périphérique de sortie :	Imprimante (basse résolution)
Résolution :	480 x 672 points d'image
Forme de ligne :	1 continue 2 petits tirets 3 pointillée 4 trait-point (ligne de symétrie) 5 longs tirets 6 trait-point-point
Epaisseurs de trait :	12
Marqueur (Symboles) :	1 . 2 + 3 * 4 o 5 x
Tailles de caractère :	12
Directions de texte :	0, 90, 180, 270, 360

Modèles de remplissage :	**Hachures :**
1	1
2	2
3	3
4	4
5	5
6	6

Elément général de représentation : barre pleine

Séquences Escape : indiquer les cellules de caractère disponibles

Entrée : non

DDFXHR8.PRL

Périphérique de sortie : Imprimante (haute résolution)

Résolution : 480 x 672 points d'image

Forme de ligne :
1 continue
2 petits tirets
3 pointillée
4 trait-point (ligne de symétrie)
5 longs tirets
6 trait-point-point

Epaisseurs de trait :	12
Marqueur (Symboles) :	1 . 2 + 3 * 4 o 5 x
Tailles de caractère :	12
Directions de texte :	0, 90, 180, 270, 360

Modèles de remplissage :

1

2

3

4

5

6

Hachures :

1

2

3

4

5

6

Elément général de représentation :	barre pleine
Séquences Escape :	indiquer les cellules de caractère disponibles
Entrée :	non

Annexe

Instructions BASIC ayant trait au graphisme

BORDER

BORDER Couleur,[,Couleur]

La surface de l'écran du CPC sur laquelle on peut écrire est entourée d'un bord (BORDER) sur lequel rien ne peut être affiché. La couleur de ce bord peut être modifiée avec l'instruction BORDER. Couleur peut être une valeur entre 0 et 26. La table suivante indique à quelles couleurs correspondent les différentes valeurs.

0 noir	14 bleu pastel
1 bleu	15 orange
2 bleu vif	16 rose
3 rouge	17 magenta pastel
4 magenta	18 vert vif
5 mauve	19 vert marin
6 rouge vif	20 turquoise vif
7 pourpre	21 vert citron
8 magenta vif	22 vert pastel
9 vert	23 turquoise pastel
10 turquoise	24 jaune vif
11 bleu ciel	25 jaune pastel
12 jaune	26 blanc brillant
13 blanc	

Vous pouvez ajouter au premier paramètre, qui est indispensable, un second code de couleur. L'instruction BORDER aura alors pour effet de faire passer la couleur du bord de l'écran de l'une à l'autre des deux couleurs. La vitesse du changement de couleur peut être fixée avec SPEED INK.

CLG

CLG [Crayon de couleur]

L'instruction CLG vide l'écran graphique. Après exécution de l'instruction CLG, l'écran graphique aura la couleur qui est affectée au crayon de couleur indiqué.

CLS

CLS [numéro de canal]

L'instruction CLS (CLear Screen) vide tout l'écran et place le curseur dans le coin supérieur gauche de la fenêtre de texte. Après exécution de CLS, la fenêtre de texte a la couleur du crayon de couleur sélectionné avec PAPER (voir CLG).

Si un numéro de canal est indiqué pour préciser l'instruction CLS, c'est la fenêtre représentée par ce numéro qui sera vidée dans les conditions décrites ci-dessus.

COPYCHR$ (pas sur le CPC 464)

COPYCHR$(numéro de canal)

La fonction COPYCHR$ permet de copier un caractère à partir d'un emplacement de l'écran de texte. Le curseur de texte doit être placé à cet effet dans une position déterminée sur l'écran, le caractère voulu doit être lu dans cette position avec COPYCHR$ et affecté à une variable de texte. Le contenu de cette variable pourra être réutilisé à volonté. Le programme d'exemple suivant illustre ce principe :

```
10 CLS
20 LOCATE 10,10
30 PRINT "*"
40 LOCATE 10,10
50 CARACTERE$=COPYCHR$(#0)
60 LOCATE 10,12
70 PRINT CARACTERE$
80 END
```

Pour cette fonction BASIC, il faut absolument indiquer à quelle fenêtre (canal) elle est censée se rapporter.

Il faut encore tenir compte d'une autre particularité de COPYCHR$. Si la position de l'écran qu'il s'agit de lire contient autre chose que le modèle de bits d'une matrice de caractère, COPYCHR$ ne pourra bien sûr identifier aucun caractère. La fonction renverra dans ce cas une chaîne vide.

CURSOR (sauf sur le CPC 464)

CURSOR [Commutateur 1][,Commutateur 2]

L'instruction BASIC CURSOR permet d'activer (1) ou de désactiver (0) les commutateurs système et utilisateur responsables de l'apparence du curseur sur l'écran. Le premier commutateur est pour le système, le second pour l'utilisateur. Le curseur est visible sur l'écran si les deux commutateurs sont activés.

DRAW

DRAW position X,position Y[,[Crayon de couleur][,Mode]]

L'instruction BASIC DRAW permet de tracer une ligne. Le point de départ de la ligne est la position actuelle du curseur graphique, son point final le point déterminé par les positions X et Y. Le paramètre optionnel de crayon de couleur permet de déterminer dans quelle couleur la ligne doit être dessinée. Le paramètre doit être compris entre 0 et 15.

Le paramètre mode (qui n'existe pas sur le CPC 464) permet de fixer l'interaction de la ligne avec le fond. Vous pouvez choisir parmi quatre modes différents :

Normal = 0
Opération XOR = 1
Opération AND = 2
Opération OR = 3

DRAWR

DRAWR distance X,distance Y[,[Crayon de couleur][,Mode]]

L'instruction DRAWR fonctionne exactement comme DRAW si ce n'est que le point final de la ligne n'est pas fixé par des coordonnées absolues mais par une distance X/Y qui est ajoutée à la position actuelle du curseur graphique.

FILL (n'existe pas sur le CPC 464)

FILL Crayon de couleur

L'instruction BASIC FILL permet de remplir n'importe quelle surface à l'intérieur de la fenêtre graphique du CPC. La couleur de remplissage utilisée est la couleur affectée au crayon de couleur.

L'opération de remplissage commence toujours dans la position actuelle du curseur graphique. Sera considérée comme limite de la surface à remplir une ligne ayant la même couleur que le crayon de couleur utilisé pour le remplissage. Sera également considérée comme ligne une ligne ayant été tracée avec le crayon de couleur défini par l'instruction GRAPHICS PEN. Si le curseur graphique figure sur une limite de ce type lors de l'appel de la routine de remplissage, l'opération de remplissage sera immédiatement interrompue.

```
10 CLS
20 MOVE 100,100
30 DRAWR 0,200
40 DRAWR 200,0
50 DRAWR 0,-200
60 DRAWR -200,0
70 MOVE 200,200
80 FILL 1
90 END
```

Ce petit programme dessine d'abord un carré (lignes 20 à 60) puis positionne le curseur graphique au milieu du carré. Les conditions sont alors remplies pour le remplissage effectué en ligne 80.

FRAME (sauf sur le CPC 464)

FRAME

Cette instruction BASIC synchronise l'écriture sur l'écran avec le retour du faisceau. Elle permet de dessiner sur l'écran sans tremblotement de l'image.

GRAPHICS PAPER (sauf sur le CPC 464)

GRAPHICS PAPER Crayon de couleur

L'instruction GRAPHICS PAPER fixe la couleur du fond de l'écran graphique. On peut indiquer un crayon de couleur entre 0 et 15. Le programme suivant montre quand le fond de l'écran graphique devient visible.

```
10 MODE 0
20 TAG
30 GRAPHICS PAPER 10
40 MOVE 100,100
50 PRINT "A";
```

Le "A" sera envoyé sur l'écran dans la couleur de caractère fixée. Les points de la matrice de caractère qui ne seront pas sortis dans la couleur de caractère pour représenter le "A" prendront la couleur du fond fixée avec GRAPHICS PAPER.

GRAPHICS PEN (sauf sur le CPC 464)

GRAPHICS PEN [Crayon de couleur][,Mode]

L'instruction BASIC GRAPHICS PEN fixe la couleur avec laquelle on dessine sur l'écran graphique. Le crayon de couleur doit être compris entre 0 et 15.

Le paramètre mode fixe si la sortie sur écran doit se faire avec fond (0) ou sans fond. Le programme suivant sort un caractère sans fond sur l'écran graphique.

```
10 MODE 0
20 TAG
30 GRAPHICS PEN 10,1
40 MOVE 100,100
50 PRINT "A";
```

En remplaçant le second paramètre par un 0, on obtiendra que le fond de caractère soit à nouveau sorti avec le caractère.

INK

INK Crayon de couleur,Code couleur[,Code couleur]

L'instruction BASIC INK permet d'affecter des couleurs aux 16 crayons de couleur possibles du CPC. Le crayon de couleur peut être toute valeur entre 0 et 15. En fonction du MODE fixé, le nombre de couleurs disponibles sera différent. En MODE 2, vous disposez de 2 couleurs (monochrome), en MODE 1 de quatre couleurs et en MODE 0 de seize couleurs.

Après mise sous tension de l'AMSTRAD CPC, les crayons de couleur sont pré-définis avec les couleurs suivantes :

Crayon de couleur	Couleur (MODE 0)	Couleur (MODE 1)	Couleur (MODE 2)
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	1,24	20	1
15	16,11	6	24

L'instruction INK permet de modifier à volonté l'affectation défaut des crayons de couleur. Si deux codes couleur sont indiqués pour l'instruction INK, la couleur du crayon de couleur défini passera de l'une à l'autre des deux couleurs indiquées. La vitesse du changement de couleur peut être fixée avec SPEED INK.

LOCATE

LOCATE[#Numéro de canal,]position X,position Y

L'instruction LOCATE permet de placer le curseur de texte dans n'importe quelle position de la fenêtre de texte. En indiquant un numéro de canal, vous pouvez spécifier que le curseur de texte doit être placé dans une fenêtre autre que la fenêtre numéro 0. Le nombre de positions Y possibles pour le curseur de texte est toujours de 25. Les positions X varient, suivant le MODE fixé, de 20, 40 à 80. L'origine des coordonnées (1,1) pour l'instruction LOCATE est dans la position Home (coin supérieur gauche) de la fenêtre de texte.

MASK (sauf sur le CPC 464)

MASK [Modèle de bits][,Commutateur]

L'instruction BASIC MASK détermine quel modèle de bits doit être utilisé pour tracer une ligne. Le modèle de bits doit être une valeur comprise entre 0 et 255. Si on convertit cette valeur en un nombre binaire (ou si on entre le nombre binaire directement comme paramètre), chaque bit mis de ce nombre représentera un point d'image mis sur une section de ligne de huit points d'image. Le modèle de la ligne sera répété tous les huit bits.

Traits :	MASK &X11110000
Points :	MASK &X10101010
Trait-point :	MASK &X11100100
Ligne pleine :	MASK &X11111111

Tant que le modèle n'a pas été modifié, le CPC dessinera systématiquement une ligne continue.

Le second paramètre de l'instruction MASK (commutateur) peut valoir 0 ou 1. Si vous entrez un zéro, le premier point d'une ligne ne sera pas dessiné ; si vous entrez un un, le premier point sera dessiné.

MODE

MODE Paramètre

L'instruction MODE fixe le mode écran. Lorsqu'une instruction MODE est entrée, toutes les fenêtres ainsi que le curseur de texte et le curseur graphique sont ramenés sur leurs valeurs défaut. Paramètre peut être 0, 1 ou 2.

MODE 0 :	20 Colonnes,	16 Couleurs
MODE 1 :	40 Colonnes,	4 Couleurs
MODE 2 :	80 Colonnes,	2 Couleurs

L'instruction MODE fixe la résolution de l'écran ainsi que le nombre de couleurs pouvant être représentées simultanément.

MOVE

MOVE Position X,Position Y[,Crayon de couleur][,Mode]

Avec l'instruction MOVE, le curseur graphique est placé sur une position absolue de l'écran graphique. Les coordonnées vers lesquelles le curseur graphique doit être déplacé sont indiquées comme paramètres de l'instruction MOVE. Si le crayon de couleur graphique actuel doit être remplacé par un autre, cela peut être obtenu en indiquant un numéro de crayon de couleur (paramètre facultatif, impossible sur le CPC 464).

Le paramètre Mode (sauf sur le CPC 464) permet de fixer l'interaction entre le crayon de couleur actuel et le fond. Vous pouvez choisir parmi quatre modes différents :

Normal	= 0
Opération XOR	= 1
Opération AND	= 2
Opération OR	= 3

MOVER

MOVER distance X,distance Y[,[Crayon de couleur][,Mode]]

L'instruction MOVER fonctionne exactement comme MOVE si ce n'est que les paramètres ne constituent pas des coordonnées absolues pour le curseur graphique mais une distance X/Y qui est ajoutée à la position actuelle du curseur graphique pour fixer la nouvelle position du curseur graphique.

ORIGIN

ORIGIN X,Y[,gauche,droite,haut,bas]

L'instruction BASIC ORIGIN fixe l'origine des coordonnées (0,0) pour la fenêtre graphique sur le point défini par les paramètres X et Y. La zone de l'écran graphique peut être délimitée par l'indication de quatre paramètres facultatifs.

PAPER

PAPER [#Numéro de canal,]Crayon de couleur

L'instruction PAPER permet de modifier la couleur du fond. Le nombre de crayons de couleur disponibles dépend du MODE activé. L'indication d'un numéro de canal, facultative, permet d'appliquer le changement de la couleur du fond à une autre fenêtre que la fenêtre numéro 0.

PEN

PEN [#Numéro de canal,][Crayon de couleur][Mode]

L'instruction PEN permet de modifier la couleur dans laquelle apparaissent les caractères sur l'écran. Le nombre de crayons de couleur disponibles dépend du MODE activé. L'indication d'un numéro de canal, facultative, permet d'appliquer le changement de la fenêtre couleur de caractère à une autre fenêtre que la numéro 0. Le paramètre mode fixe si la sortie sur écran doit être transparente (1) ou non (0).

PLOT

PLOT Position X,Position Y[,Crayon de couleur][,Mode]

L'instruction PLOT permet de placer le curseur graphique sur une position absolue de l'écran graphique et de sortir un point sur cette position. Si le crayon de couleur graphique actuel doit être remplacé par un autre, cela peut être obtenu en indiquant un numéro de crayon de couleur (paramètre facultatif, impossible sur le CPC 464).

Le paramètre Mode (sauf sur le CPC 464) permet de fixer l'interaction entre le crayon de couleur actuel et le fond. Vous pouvez choisir parmi quatre modes différents :

Normal	= 0
Opération XOR	= 1
Opération AND	= 2
Opération OR	= 3

PLOTR

PLOTR distance X,distance Y[,[Crayon de couleur][,Mode]]

L'instruction PLOTR fonctionne exactement comme PLOT si ce n'est que les paramètres ne constituent pas des coordonnées absolues pour la sortie du point mais une distance X/Y qui est ajoutée à la position actuelle du curseur graphique pour fixer la nouvelle position où le point sera sorti.

POS

POS(#Numéro de canal)

La fonction BASIC POS fournit la coordonnée X actuelle (par rapport au bord gauche de l'écran) de la position du curseur de texte. Le numéro de canal doit absolument être indiqué comme paramètre de cette fonction.

SPEED INK

SPEED INK Durée 1,Durée 2

Les instructions BASIC ORDER et INK permettent de définir deux couleurs qui apparaîtront par alternance dans un emplacement colorié. La vitesse du changement de couleur peut être modifiée avec l'instruction SPEED INK. Le paramètre durée 1 fixe combien de temps la première couleur doit rester visible alors que le paramètre 2 fixe la durée pour la seconde couleur. Les durées sont fixées en unités d'un cinquantième de seconde.

SYMBOL

SYMBOL Code caractère,Matrice de caractère

L'instruction BASIC SYMBOL attend neuf paramètres dont les valeurs doivent être comprises entre 0 et 255.

La première valeur suivant SYMBOL (le code caractère) fixe quel caractère sera remplacé par une nouvelle matrice définie par l'utilisateur. Les huit autres paramètres définissent cette matrice.

```
10 SYMBOL AFTER 123
20 SYMBOL 123,&X11011011,&X00111100,&X01100110,&X01100110,&X011111
10,&X01100110,&X01100110,&X00000000
```

Ce programme redéfinit le caractère 123 ({) en un "Ä" à l'aide de l'instruction SYMBOL. Nous avons choisi d'entrer les paramètres en notation binaire pour faire ressortir le lien entre les nombres et la matrice de caractère. Si un bit est mis dans un des huit derniers paramètres, le bit correspondant sera également mis dans la matrice de caractère (voir CHR$(25)).

Notez bien qu'avant d'utiliser l'instruction SYMBOL, vous devez autoriser le redéfinition à partir du code de caractère voulu avec SYMBOL AFTER.

SYMBOL AFTER

SYMBOL AFTER Paramètre

L'instruction SYMBOL AFTER fixe quels caractères peuvent être redéfinis par l'utilisateur. Le paramètre doit être compris entre 0 et 255. Il indique le code de caractère à partir duquel (inclusivement) tous les caractères pourront être remplacés par des matrices de caractère définies par l'utilisateur avec SYMBOL. Si le paramètre suivant SYMBOL AFTER est 256, aucun nouveau caractère ne peut être défini.

TAG

TAG [#Numéro de canal]

L'instruction BASIC TAG a pour effet que la sortie de caractères de texte ne se fera plus désormais sur la position du curseur de texte mais dans celle du curseur graphique. Le texte sera sorti de telle façon que la position la plus élevée d'une matrice de caractère coïncide avec les coordonnées du curseur graphique. L'indication (facultative) d'un numéro de canal permet de fixer sur quel canal (quelle fenêtre) les caractères de texte devront être sortis dans la position du curseur graphique.

Si une sortie de texte avec PRINT est effectuée après un TAG, elle doit se terminer par un ";" pour éviter que des caractères de commande ne soient sortis comme caractères de texte.

TAGOFF

TAGOFF [#Numéro de canal]

L'instruction BASIC TAGOFF annule l'effet de TAG pour le numéro de canal (facultatif) indiqué. La sortie de texte se fera donc à nouveau dans la position du curseur de texte.

TEST

TEST(Position X,Position Y)

La fonction TEST fixe le curseur graphique sur la position de l'écran indiquée par les paramètres. TEST fournit ensuite le numéro du crayon de couleur qui a été utilisé dans cette position.

TESTR

TESTR(Distance X,Distance Y)

La fonction TESTR positionne le curseur graphique en fonction des distances X et Y indiquées, par rapport à sa position de départ. TESTR fournit ensuite le numéro du crayon de couleur qui a été utilisé dans cette position.

VPOS

VPOS(#Numéro de canal)

La fonction BASIC POS fournit la coordonnée X actuelle (par rapport au bord gauche de l'écran) de la position du curseur de texte. Le numéro de canal doit absolument être indiqué comme paramètre de cette fonction.

WINDOW

WINDOW [#Numéro de canal,]gauche,droite,haut,bas

L'instruction BASIC WINDOW définit les dimensions d'une fenêtre. Comme les coordonnées à entrer sont des coordonnées de l'écran de texte, la valeur indiquée pour les colonnes doit tenir compte du MODE fixé. Si aucun numéro de canal n'est précisé, l'instruction se rapporte automatiquement au canal numéro 0.

WINDOW SWAP

WINDOW SWAP Numéro de canal,Numéro de canal

L'instruction WINDOW SWAP échange entre elles les zones de l'écran (les fenêtres) définies par les deux numéros de canal. La syntaxe de WINDOW SWAP exige que ces numéros soient indiqués sans être précédés de "#".

XPOS

XPOS

La fonction BASIC XPOS fournit la coordonnée X actuelle de la position du curseur graphique.

YPOS

YPOS

La fonction BASIC YPOS fournit la coordonnée Y actuelle de la position du curseur graphique.

Lexique

Accumulateur (Accu)
Registre de l'unité centrale dans lequel sont placés les résultats d'opérations arithmétiques, logiques et d'entrée/sortie.

Adresse
Emplacement de la mémoire qui est généralement indiqué sous forme d'un nombre hexadécimal sur deux octets. La zone d'adresses (0 à 32767) est représentée en hexadécimal par [0000 to FFFF] [-32768,...,-1].

Adresse de base
L'adresse de base plus une distance (offset ou adresse relative) permettent de calculer une adresse absolue.

Algorithme (Procédure de calcul)
Traitement étape par étape de la solution d'un problème déterminé. Un algorithme peut être exprimé par n'importe quels symbole ou élément de langage.

Alphabétique
Désigne strictement les lettres A à Z.

Alphanumérique
Désigne les lettres A à Z et les chiffres 0 à 9.

Argument
Chaîne de caractères ou nombre envoyé à une fonction puis traité par celle-ci pour fournir un résultat. Ce résultat est la valeur de fonction.

ASCII (American Standard Code for Information Interchange)
Norme internationale de code 7 bits utilisée pour le stockage de données de texte et l'échange de données.

Assembleur
Langage de programmation permettant de représenter le code machine binaire par des abréviations (mnémoniques).

BASIC
Abréviation de Beginners' All purpose Symbolic Instruction Code.

Binaire (binary)
N'autorise que deux représentations, 0 ou 1. Le système numérique binaire (système de base 2) utilise uniquement les uns et les zéros pour représenter toutes sortes de grandeurs. Cela correspond au mode de représentation interne des données sur l'ordinateur puisque celui-ci ne connaît que deux états fondamentaux qui correspondent à 0 et à 1.

Bit
Abréviation de Binary Digit. Un bit est la plus petite unité de mémoire de l'ordinateur. Il peut représenter les valeurs 0 ou 1.

Buffer
Zone réservée, dans l'ordinateur, au stockage provisoire de données.

Canal
Le canal se rapporte aux entrées/sorties de l'AMSTRAD CPC. Le numéro de canal permet d'appeler une interface d'entrée/sortie déterminée.

Caractère
Elément d'image produit par l'ordinateur. Les lettres et les chiffres sont les caractères les plus utilisés.

Caractères spéciaux
En traitement de texte, on qualifie de caractères spéciaux tous les caractères, par exemple "+", "@", "#", qui ne sont ni des lettres ni des chiffres.

Compatibilité
Possibilité d'interchanger des éléments d'un système ou d'utiliser des programmes sur d'autres systèmes sans modification.

Curseur
Marque sur l'écran la prochaine position d'écriture.

Décimal
Permet 10 représentations de chiffres, 0 à 9. Le système décimal (système de base 10) est celui que nous utilisons tous les jours. L'AMSTRAD CPC stocke les nombres décimaux en code binaire.

Défaut
Action ou valeur effectuée ou fournie par le programme si vous n'avez pas précisé quelle action doit être effectuée ou quelle valeur doit être fournie.

Désassembleur
Programme traduisant un code hexadécimal en représentation mnémonique et calculant la bonne longueur d'instruction.

Device
(voir périphérique)

Disquette
Support magnétique de stockage de masses de données importantes.

Données
Informations transmises à un programme ou fournies par un programme. Il y a quatre sortes de données :

- Nombres entiers
- Nombres en précision simple
- Nombres en double précision
- Chaînes de caractères (texte)

Edition/traitement
Opération consistant à examiner et à modifier les informations existantes.

Graphisme avec caractères
Graphisme informatique composé d'une multitude de caractères et caractères spéciaux produits par le générateur de caractères de l'ordinateur.

Graphisme de grille
Méthode consistant à diviser l'écran en une quantité de points déterminée de façon à ce que chaque point puisse être modifié (allumé ou éteint) séparément.

Le dessin est le produit de la manipulation de l'ensemble des points de la grille.

Graphisme vectoriel
Le graphisme vectoriel est un procédé qui permet de représenter des corps en trois dimensions sur un ordinateur. Les corps produits à l'aide du graphisme vectoriel sont constitués d'une multitude de lignes symbolisant les côtés de l'objet.

Hexadécimal (Hexa)
Le format hexadécimal permet 16 représentations de chiffres. Les chiffres hexadécimaux sont représentés par 0, 1, 2,..., 9, A, B, C, D, E, F. Les nombres hexadécimaux (base 16) sont constitués par des suites de chiffres hexadécimaux. Les valeurs d'adresse et d'octet sont souvent indiquées sous forme hexadécimale. L'AMSTRAD CPC vous permet d'entrer des constantes hexadécimales. Elles doivent être précédées d'un "&".

Incrément
Valeur d'augmentation du compteur chaque fois qu'est achevée une procédure devant être répétée.

Interpréteur
Par exemple interpréteur BASIC, traduit et exécute ligne par ligne, sous forme d'instructions en langage machine, les instructions de programmation écrites dans un langage évolué.

Interruption (Break)
Interrompt l'exécution d'un programme. Une instruction STOP entraînera en BASIC l'interruption du programme, de même que la touche BREAK.

Kilo octets ou K
Représente 1024 octets de mémoire. Un système de 64 K a donc une capacité de 64 x 1024 = 65536 octets.

Langage machine
Langage compris et exécuté par un microprocesseur comme, par exemple, le jeu d'instructions du Z80. Ce langage est en général représenté en code hexadécimal. Tous les langages de programmation évolués doivent être traduits ou interprétés en langage machine pour pouvoir être exécutés par l'ordinateur.

Mémoire buffer (Buffer)
Zone de la RAM dans laquelle sont accumulées des données en vue d'un traitement ultérieur.

Mémoire de travail
(voir RAM)

Menu
Table offrant plusieurs possibilités de choix à l'utilisateur et lui indiquant comment opérer sa sélection.

Mnémoniques
Les mnémoniques sont des abréviations représentant des codes binaires et en évoquant la signification.

Modèle d'armature
Objets tridimensionnels produits à l'aide du graphisme vectoriel.

Octet
Plus petite unité de mémoire adressable sur l'ordinateur. Comprend 8 bits et peut donc représenter 256 valeurs différentes, c'est-à-dire, exprimé en décimal, les valeurs 0 à 255.

Octet faible
Octet le moins significatif d'une valeur sur deux octets (Least Significant Byte ou LSB).

Octet fort
Octet le plus significatif d'une valeur sur deux octets (Most Significant Byte ou MSB).

Paramètre
Informations fournies avec une instruction pour en préciser l'effet voulu.

Périphérie
Toutes les machines d'entrée et de sortie, mémoires et mémoires auxiliaires, placées en dehors de l'ordinateur et collaborant avec lui.

Périphérique (Device)
Elément physique d'un système informatique qui est utilisé pour l'entrée/sortie de données, par exemple le clavier, l'écran ou l'imprimante.

Point d'image
Point de l'écran graphique qui peut être visible ou invisible.

Quartet
Nom donné à un groupe de quatre bits (demi-octet).

Quartet faible
Moitié de plus faible poids d'un octet.

Quartet fort
Moitié de plus fort poids d'un octet.

RAM (Random Access Memory)
Mémoire de lecture/écriture avec accès sélectif.

ROM (Read Only Memory)
Mémoire de lecture seulement, qui conserve son contenu même après interruption de l'alimentation électrique.

Routine/sous-programme
Suite d'instructions chargée de l'exécution d'une fonction déterminée. Une routine est souvent un sous-programme qui sera appelé en plusieurs endroits du programme principal.

Statement
Mot anglais, utilisé par l'ordinateur dans les messages d'erreur, signifiant "instruction".

Syntaxe
Le terme de syntaxe est utilisé pour désigner les règles "grammaticales" de l'emploi d'une instruction. La syntaxe fixe généralement les caractères et la séquence d'éléments à utiliser dans une instruction.

Tableau
Jeu organisé d'éléments de données pouvant être appelés ensemble ou séparément au moyen du nom du tableau et d'un ou plusieurs indices. En BASIC, tout nom de variable peut être utilisé pour désigner un tableau. Les tableaux peuvent avoir une ou plusieurs dimensions. AR(x) représente un tableau à une dimension appelé AR. AR(x,y) représente un tableau à deux dimensions appelé AR, etc.

Vecteur

1. Adresse de mémoire programme qui est communiquée à l'ordinateur indépendamment de tout résultat déterminé.
2. Droite orientée et située dans l'espace.

Wire Frame
(voir modèle d'armature).

INDEX

A

Accents 85
Accès à la RAM 280
Accu 16 bits 283
Accumulateur 281
Adresse de base 427
Adresse de fonction BDOS 461
Adresse écran 289
Adresse en page zéro 463
Adresses de départ 464
Adresses de destination 463
Adresses de distance 283
Affectation univoque 187
Affichage vidéo 456
Affichages d'erreur 72
Algorithme de ligne 258
Algorithmes 3D 256
Angle d'observation 221
Animation 1, 215, 225, 229, 240, 242
Animation d'objet 218
Appel de fonction BDOS 468
Appel rapide 20
Appels de GSX 458
Argument 187
Armature 215
ASCII 33
ASSIGN.SYS 457, 459
Axe des X 189
Axe des Y 190
Axe des Z 216, 436
Axé sur une grille 14, 6

B

BANKMAN 132
Banque 446
Banque de RAM 291
BAR 479
Bascule bas-haut 264
BASIC Locomotive 13, 70
BASIC Mallard 80 458
BASIC.COM 461
Binaire 34
Bit 1, 33
Bloc de dimensionnement 218
BLocs de paramètres 464
BORDER 64
Buffer graphique 474

C

Calculatrice de poche 426
Capacité mémoire 14
Caractère 33
Caractère ASCII 34
Caractère écran 14
Caractères d'impression 273
Caractères de commande 34
Caractères de commande BASIC 52
Caractères spéciaux 14, 34
Cercle 13
Cercle fermé de 360 degrés 176
Chaîne de caractères 425
Chiffre d'affaire 147
Choix du menu 263, 266
Clarté de couleur 257
Clavier 259
CLG 473
CLOSEDR 472
Code 33
Code d'opération 283
Code de caractère 33

Codé en bits 259
Codes ASCII 8 bits 36
Communication 33
Commutateur de seuil 264
Compatibilité 305
Compilateur 424
Compteur d'index de variable 466
Compteur de programme 283
Configuration de la mémoire 37, 291
Configuration logicielle 491
Connexion d'imprimante 7 bits 269
Constante 465
Contenu de l'écran 230, 267
Contenu de surface 217
Contours 226
CONTROL 464
Contrôleur vidéo 264, 285, 448
Coordonnée 465
Coordonnées 3D 225
Coordonnées de test 305
Copie d'écran 131, 163, 267
Copie d'écran de texte 267
Copie d'écran graphique 267
Côtés cachés 256
Couleur du cadre 290
CP/M 460
CPC-6128 456
CPC-Chart 160
CPCs World 231
Crayon optique 262, 285, 289
Crayons de couleur 19
Curseur 54, 55
Curseur d'édition 71
Curseur de texte 9
Curseur graphique 9, 160
Cycle d'horloge 292, 299

D

DD-DMP1.PRL 458
DDFXHR8.PRL 457, 510

DDFXLR7.PRL 458
DDFXLR8.PRL 457, 509
DDHP7470.PRL 457
DDMODE0.PRL 458
DDMODE1.PRL 458
DDMODE2.PRL 458
DDSCREEN.PRL 457, 508
DDSHINWA.PRL 458
Définition 187
Définition d'affectation 197
Définition de caractère 116
Définition de Sprite 426
Demi-image 14
Démo graphique 132
Désassembleur 334, 491
DESTROYED 116
DIGITAL RESEARCH 455
Digitaliser 64
Disquette 131
Disquettes système 456
Domaine de définition 191
Domaine de valeurs 188, 427
Données de Sprite 425, 426
DRIVERS.GSX 457
Droites 188

E

Echelle 150
Echelle de valeurs 156
Ecran 262
Ecran couleur 266
Editeur d'animation 237
Editeur de jeu de caractères 69
Editeur de lien 236
Editeur de sommets d'angle 234
Editeur de Sprite 426, 442
Editeur graphique 120
Effet de dessin animé 445
Electronique 37
Elément 187

Elément de table 259
Elément fictif 468
Eléments de matrice 217
Eléments graphiques 5
Ellipses 14
Ensemble de nombres 188
EPSON 268, 458
Equation de fonction 188
Extension de la mémoire 259
Extension GSX 455

F

Facteur d'agrandissement 192
Fenêtre d'option 20
Fenêtre de dialogue 20
Fenêtre écran 192
Feuille de dessin 20
Fichier binaire 132
FILLCOLOR 490
FILLINDEX 489
FILLPOLY 478
FILLSTYLE 488
Films électroniques 215
Flux de nombres 188
Fonction aléatoire 121
Fonction d'angle 229
Fonction Fill 257
Fonction quadratique 200
Fonctions 187
Fonctions linéaires 196
Fond 437
Fréquence de répétition de l'image 287

G

Gate Array 289
GDOS 456
Générateur de caractères 36
Génération de Sprite 425

GENGRAF BASIC 459
GENGRAF.COM 457, 461
Gestion du curseur 287
GETPAR 465
GETPOS 465
GIOS 456
GRA ASK CURSOR 318
GRA CLEAR WINDOW 323
GRA CONVERT POS 334
GRA EXTENDED INITIALISE 331
GRA FILL 335
GRA FIRST POINT 332
GRA GET ORIGIN 319
GRA GET PAPER 325
GRA GET PEN 324
GRA GET W HEIGHT 322
GRA GET W WIDTH 321
GRA INITIALISE 316
GRA LINE ABSOLUTE 329
GRA LINE RELATIVE 329
GRA MOVE ABSOLUTE 317
GRA MOVE RELATIVE 318
GRA PLOT ABSOLUTE 326
GRA PLOT RELATIVE 326
GRA RESET 316
GRA SET MASK 333
GRA SET ORIGIN 319
GRA SET PAPER 325
GRA SET PEN 323
GRA TEST ABSOLUTE 327
GRA TEST RELATIVE 328
GRA TRANS SWITCH 332
GRA WIN HEIGHT 321
GRA WIN WIDTH 320
GRA WR CHAR 330
Graduation 190
Grandeurs absolues 177
Graphe 200
Graphique camembert 160, 176
Graphique de points 149, 160
Graphiques d'entreprise 147
Graphisme de grille 7, 424

Graphisme électronique 14
Graphqe de fonction 187, 196
Gros ordinateurs 14
GSX 455
GSX.SYS 457, 458, 461

H

HD 6845 285
Hexadécimal 34
HIMEM 469
Histogramme 455
Histogrammes 14, 148, 160

I

IBM-PC 5
Idée de jeu 116
IM2 282
Image de moniteur 286
Images 120
Images électroniques 215
Implantation de GSX 458
Imprimante 14, 267
Imprimante matricielle 456
Impulsion 262
Impulsion de synchronisation 449
Impulsion HSync 286
Impulsion VSync 287
Informations d'image 5
Informations de couleur 5
INK 64
Instruction BDOS 461
Instruction CALL 466
Instruction CHR$ 39
Instruction CP/M-Plus 455, 460
Instruction de transfert de bloc 446
Instruction DEFINT 466
Instruction DRAW 12, 131, 475
Instruction LDIR 230, 299

Instruction MOVE 10
Instruction TAG 425
Instruction USR 466
Instructions BANK 445
Instructions graphiques relatives 10
Interface 259
Interférences 14
Interlace 287
Interruption 294, 299
Interruption du programme 506

J

Jeu d'arcade 115
Jeu d'instructions GSX 469
Jeu de caractères 33, 52
Jeu de caractères alternatif 89
Jeu de guerre des étoiles 120
Jeu électronique 216
Jeu vidéo 115
Jeux de registres principaux 292
Jeux vidéo 115
JOY 259
JOYCE 455
Joystick 259

L

Langage machine 37, 424
Langage machine 8080 463
Langages évolués 424
Lettres 34
Ligne 11
Ligne de grille 287
Lignes cachées 256
LINECOLOR 483
LINESTYLE 482
Listing de GDOS 491
Listing ROM 305
Littérature spécialisée sur le Z80 280

LOADDR 470
LOCATE 63

M

Manipulation de couleur 292
Masques bits 303
Matrice 69, 216
Matrice de caractère 36, 70
Mémoire de travail 14
Mémoire écran 5, 132, 446
Mémoire principale 425
Mémoire RAM 268
MEMORY 469
Memory Adress Lines 264
MERGE 120
Messages d'erreur GSX 506
Microprocesseur 491
MKRCOLOR 486
MKRSIZE 485
MKRSTYLE 484
MODE 6, 36, 56, 297
Mode d'écriture graphique 61
Mode graphique 5
Mode jeu vidéo 115
Mode transparent 60
Mode XOR 425
Modèle bits 36
Modèle d'armature 231, 232
Modèle de surface pleine 215
Modifications d'objet 217
Module de lignes cachées 256
Modules d'extension 255
Modules GSX 457
Moniteur de données 14
Monochrome 6, 18
Mosaïque 1
Mot de commande GDOS 468
Mot de données 285

N

Niveau TTL 264
NLQ 401 268
Nombres aléatoires 282
Nombres réels 188
Norme de caractère 33
Nuances de gris 65
Numéro de Sprite 426
Numéros de périphérique 457

O

Octets de commande 436
Offset 225, 283
Ombre 257
Opération 8 bits 281
OPTION BASE 1 466
Ordinateurs domestiques 89
Organisation de la mémoire de GSX 462
Organisation des données 465
Organisation du processeur 279
Origine des coordonnées 427
OUTPUT 474

P

Page écran 229, 441, 445
Page graphique 1
PAPER 59, 62
Papier continu 273
Parabole 201
Parabole normale 201
Paramètre de décalage 452
PC 14
PEN 59, 62
Pénombre 257
Périphérique 259, 507
Périphérique de sortie de données 14

Périphérique de visualisation des données 14
Perspective 216
Perspective centrale 227, 228
Perspective parallèle 220
Phase d'initialisation 463
Phosphore 14
Phototransistor 263
Place mémoire 14, 5
Plantage du programme 463
Plantage du système 293
PLOT 7, 131
Plotter 456
Plotter de fonction 203
Point de fuite 220, 226
Pointeur de joystick 260
Pointeur de pile 283
Points d'image 7
Points du menu 232
Polygone 14
Porte logique 264
Portions d'image 453
Ports d'entrée/sortie 285
Position écran 426
Position HOME 6
Possibilités graphiques 131
Pourcentages 177
Processeur 463
Processeur 8 bits 280
Processeur de périphérique 456
Programmation de jeux 449
Programmation graphique 293
Programmation machine 279
Programme de chargement BASIC 268
Programme de chargement de GSX 461
Programme de dessin 19
Programme de support 436
Programmes de dessin 120
Programmes graphiques 131
Projection centrale 226
Projection d'objet 228
Projection parallèle 220
PUSH 299

Q

Quartet 303
QWERTY 86

R

Rayon cathodique 14, 262, 441
Rectangle 13
Refresh 281
Registre 280
Registre 16 bits 282, 295
Registre 8 bits 281
Registre d'adresse 285
Registre d'état 281
Registre d'index 283
Registre d'interruptions 282
Registre de crayon optique 264
Registre de données 285
Registre double 282
Registre Flag 281
Registre INK 290
Registre PEN 290
Registre refresh 281
Registre spécial 281
Registre multifonctions 290
Relogeabilité 463
Représentation de texte 5
Représentation graphique 5
Résistance 264
Résolution 5, 8
Résolution arithmétique Y 8
Résolution graphique 456
Résolution Y réelle 8
Retour du faisceau 441
RND 121
ROM 36
Rotation 225
Rotation d'objet 217
Routine de scrolling 449

Routines graphiques 305
Routines ROM 305
RPED 459

S

Scénarios 118
SCR DOT POSITION 313
SCR HORIZONTAL 314
SCR PIXELS (Force Mode) 314
SCR VERTICAL 315
SCREENCOPY 229, 445
SCREENSWAP 229, 445
Scrolling 449
Scrolling horizontal 452
Scrolling vertical 452
Second jeu de registres 292
Second registre 294
Section de la ROM 334
Séquence d'animation 231, 241, 255
Séquence de commande 270
SETMKR 476
SETPAR 465
SETPOS 465
Signal sonore ASCII 58
Simulation d'ombre 257
Sommets d'angle 218
Source lumineuse 257
Sources de lumière 257
Sprites 424, 425
Standard ASCII 34
Standard DIN 86
Studios graphiques 215
Style de programmation 492
SYMBOL 63
Synchronisation 14
Système d'exploitation 13, 267
Système de coordonnées 158, 189, 216
Système de coordonnées 3D 216
Système graphique vectoriel 14

T

Table 300, 259
Table ASCII 34
Table d'octets faibles 301
Table d'octets forts 301
Table de valeurs 188
Table de vecteurs 305
Tables de fonction d'angle 259
Technique de quart de cercle 15
Temps de calcul 425
Terminal de texte 89
TEXT 477
Tolérances 507
Touche d'entrée 58
Traitement de données 33
Transfert de données 280
Transformation 221, 225
Transistor Darlington 264
Transmission de coordonnées 465
Transmission de paramètres 466
Tremblotements 437
Triangle 13
Tridimensionnel 215, 231
Trimmer 265, 266
TRS-80 14
TTL 264
Tube cathodique 262
Tubes de régénération de l'image 14
TXT GET CURSOR 312
TXT GET WINDOW 311
TXT INITIALISE 308
TXT OUTPUT 309
TXT RESET 308
TXT SET CURSOR 312
TXT SET GRAPHIC 310
TXT WIN ENABLE 310
TXTCOLOR 487
TXTDIR 481
TXTSIZE 480
Type de variable 464

U

Unité centrale 280, 285
Unité centrale 8080 280
Unité centrale Z80 280
Utilitaire 335

V

Valeur de fonction 187
Variable 187
Variables entières 466
Vecteur d'interruptions 291
Vecteur Jump 447
Vecteurs 14, 217, 292
Vitesse d'animation 425
Vitesse de projection 220
Vitesse de traitement 202

W

WINDOW 63
Wire-Frame 215

Z

ZILOG 280
Zone ROM 291

les livres Amstrad



TRUCS ET ASTUCES POUR L'AMSTRAD CPC (Tome 1)

C'est le livre que tout utilisateur d'un CPC doit posseder. De nombreux domaines sont couverts (graphismes, fenetres, langage machine) et des super programmes sont inclus dans ce best-seller (gestion de fichiers, editeur de textes et de sons...)
Ref. : ML 112
Prix : **149** FF



PROGRAMMES BASIC POUR LES CPC (Tome 2)

Alimentez votre CPC. Ce livre contient de supers programmes, notamment un desassembleur, un editeur graphique, un editeur de texte... Tous les programmes sont prets a etre tapes et abondamment commentes.
Ref. : ML 118
Prix : **129** FF

LE BASIC AU BOUT DES DOIGTS CPC (Tome 3)

Ce livre est une introduction complete et didactique au BASIC du micro-ordinateur AMSTRAD CPC 464. Il permet d'apprendre rapidement et facilement la programmation (instructions BASIC, analyses des problemes, algorithmes complexes...). Comprenant de nombreux exemples, ce livre vous assure un apprentissage simple et efficace du BASIC CPC.
Ref. : ML 119
Prix : **149** FF



AMSTRAD OUVRE-TOI (Tome 4)

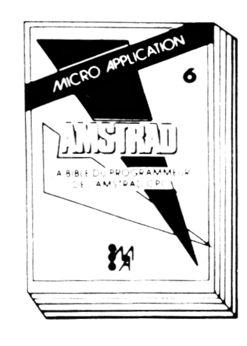
Le bon depart avec le CPC 464 ! Ce livre vous apporte les principales informations sur l'utilisation, les possibilites de connexions du CPC 464 et les rudiments necessaires pour developper vos propres programmes. C'est le livre ideal de tous ceux qui veulent penetrer dans l'univers des micro-ordinateurs avec le CPC.
Ref. : ML 120
Prix : **99** FF





JEUX D'AVENTURES. COMMENT LES PROGRAMMER (Tome 5)

Voici la cle du monde de l'aventure. Ce livre fournit un systeme d'aventures complet, avec editeur, interpreteur, routines utilitaires et fichiers de jeux. Ainsi qu'un generateur d'aventures pour programmer vous-memes facilement vos jeux d'aventures. Avec, bien sur, des programmes tout prets a etre tapes.
Ref. : ML 121
Prix : **129** FF



LA BIBLE DU PROGRAMMEUR DE L'AMSTRAD CPC (Tome 6)

Tout, absolument tout sur le CPC 464. Ce livre est l'ouvrage de reference pour tous ceux qui veulent programmer en pro leur CPC. Organisation de la memoire, le controleur video, les interfaces, l'interpreteur et toute la ROM DESASSEMBLEE et COMMENTEE sont quelques-uns des themes de cet ouvrage de 700 pages.
Ref. : ML 122
Prix : **249** FF

les plus de Micro Application



LE LANGAGE MACHINE DE L'AMSTRAD CPC (Tome 7)

Ce livre est destiné a tous ceux qui désirent aller plus loin que le BASIC. Des bases de la programmation en assembleur a l'utilisation des routines système, tout est expliqué avec de nombreux exemples. Contient un programme assembleur, moniteur et désassembleur.

Réf. : ML 123
Prix : **129** FF



GRAPHISMES ET SONS DU CPC (Tome 8)

L'AMSTRAD CPC dispose de capacites graphiques et sonores exceptionnelles. Ce livre en montre l'utilisation a l'aide de nombreux programmes utilitaires.

Ref. : ML 124
Prix : **129** FF

PEEKS ET POKES DU CPC (Tome 9)

Comment exploiter a fond son CPC a partir du BASIC? C'est ce que vous revele ce livre avec tout ce qu'il faut savoir sur les peeks, pokes et autres call... Vous saurez aussi comment proteger la memoire, calculer en binaire... et tout cela tres facilement. Un passage assure et sans douleur du BASIC au puissant LANGAGE MACHINE.

Ref. : ML 126
Prix : **99** FF



LIVRE DU LECTEUR DE DISQUETTE AMSTRAD CPC (Tome 10)

Tout sur la programmation et la gestion des donnees avec le 6128 DDI-1 et le 664! Utile au debutant comme au programmeur en langage machine. Contient le listing du DOS commente, un utilitaire qui ajoute les fichiers RELATIFS a l'AMDOS avec de nouvelles commandes BASIC, un MONITEUR disque et beaucoup d'autres programmes et astuces.



Ref. : ML 127
Prix : **149** FF



MONTAGES, EXTENSIONS ET PERIPHERIQUES AMSTRAD CPC (Tome 11)

Pour tous les amateurs d'électronique, ce livre montre ce que l'on peut realiser avec un CPC. De nombreux schémas et exemples illustrent les thèmes et applications abordés comme les interfaces, programmateur d'EPROM... Un très beau livre de 450 pages.

Réf. : ML 131
Prix : **199** FF



LE LIVRE DU CP/M AMSTRAD (Tome 12)

Ce livre vous permettra d'utiliser CP/M sur les CPC 464, 664 et 6128 sans aucune difficulté. Vous y trouverez de nombreuses explications et les différents exemples vous assureront une maitrise parfaite de ce très puissant système d'exploitation qu'est CP/M.

Réf. : ML 128
Prix : **149** FF

les livres Amstrad



DES IDEES POUR LES CPC (Tome 13)

Vous n'avez pas d'idees pour utiliser votre CPC (464, 664, 6128)? Ce livre va vous en donner! Vous trouverez de tres nombreux programmes BASIC couvrant des sujets tres variés qui transformeront votre CPC en un bon petit genie. De plus les programmes vous permettront d'approfondir vos connaissances en programmation.

Ref : ML 132
Prix : **129** FF



LES ROUTINES DE L'AMSTRAD CPC (Tome 14)

Pour bien connaitre et utiliser les routines utiles de l'AMSTRAD 6128, 664, 464. A la portee de tous. Nombreux programmes utilitaires, exemples, desassembleur, etc.

Ref : ML 143
Prix : **149** FF

DÉBUTER AVEC LE CPC 6128 (tome 15)

Ce livre s'adresse a ceux qui debutent avec le CPC 6128. Tout leur est clairement explique aussi bien pour le materiel que pour le logiciel. Une fois leur machine bien en main, ils pourront s'attaquer au Basic et utiliser l'utilitaire de gestion d'adresses que contient le livre.

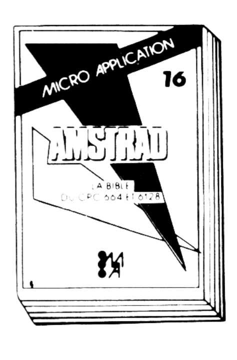
Réf : ML 145
Prix : **99** F TTC

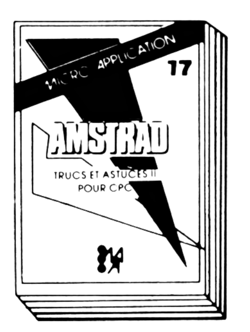


LA BIBLE DES CPC 664/6128 (tome 16)

Un regal pour tous ceux qui veulent tout connaitre sur les CPC 6128 et 664. Analyse du systéme d'exploitation, du processeur, le GATE ARRAY, le contrôleur video, le 8255, le chip sonore, les interfaces. Comprend un desassembleur, les points d'entrée des routines commentes de l'interpreteur et du systeme d'exploitation. Un super livre comme toutes les Bibles !

Réf : ML 146   Prix : **199** F

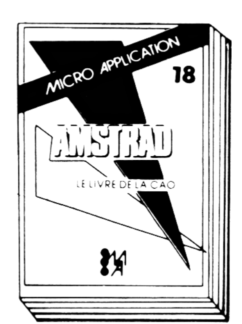




TRUCS ET ASTUCES II POUR CPC (tome 17)

Ce livre concerne tous les possesseurs de CPC (464, 664 et bien sûr 6128 !). Vous y trouverez un generateur de menus, un generateur de masques, des aides a la programmation comme un DUMP, l'utilisation des routines systemes et plein d'astuces de programmation. Pour tous ceux qui veulent tirer le maximum de leur CPC !

Réf : ML 147
Prix : **129** F TTC



Ref : ML 148
Prix : **149** FF
Disponible en Mai

LE LIVRE DE LA CAO (Tome 18)

Avec cet ouvrage vous saurez tout sur la Conception Assistee par Ordinateur et sur la programmation des GRAPHIQUES en 3 dimensions sur les CPC. Les points, lignes, rectangles, cercles, courbes, figures en 3D (comme les cubes, pyramides, cylindres, etc.), les rotations, les effets miroirs, les eclatements et explosions, et enfin pour conclure le clou : toutes les astuces pour creer son propre systeme de CAO. Nombreux programmes exemples et utilitaires.

les plus de Micro Application



PROGRAMMES et APPLICATIONS ÉDUCATIFS sur CPC. (Tome 19)

Ce livre est un recueil complet de programmes complets et d'applications prêts à fonctionner sur CPC. Chaque programme est très bien commenté et l'ouvrage couvre de nombreux sujets (mathématiques, chimie...). Ce livre est tout particulièrement destiné aux lycéens.

Réf : ML 150
Prix : **179** FF



Disponible en Mai

SYSTÈMES DE TRANSMISSION SUR CPC. (Tome 20)

Encore une exclusivité Micro Application. Grâce à ce livre les communications et transmissions n'auront plus de secrets pour vous et vous pourrez profiter au maximum des possibilités offertes aujourd'hui dans ce domaine. Complet avec beaucoup d'applications pratiques, un ouvrage pratique et original.

Réf : ML 151
Prix : **199** F

LE LIVRE DU LOGO (Tome 21)

Le LOGO est un langage très intéressant dont les applications sont très nombreuses. Cet ouvrage permettra au lecteur de profiter au maximum du LOGO livré avec L'AMSTRAD. Principaux thèmes abordés : les graphismes, les procédures, les récursions, les routines de tri, un générateur de masque, structure des données, intelligence artificielle...

Ref. : ML 162
Prix : **149** FF
Disponible en Juin

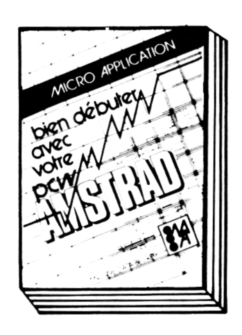


INTELLIGENCE ARTIFICIELLE ET ROBOTIQUE SUR CPC (Tome 22)

Ce livre est une excellente introduction au monde de l'intelligence artificielle et à ses applications. Toutes les techniques et méthodes décrites sont illustrées de programmes exemples. On apprendra ainsi quelle méthode un robot utilise pour trouver la sortie d'un labyrinthe ou comment un ordinateur peut acquérir des connaissances et ainsi aider à la résolution de problèmes.



Ref. : ML 163
Prix : **149** FF
Disponible en Juin



BIEN DEBUTER AVEC LE PCW

Le premier livre pour l'AMSTRAD PCW ! Cet ouvrage vous permettra de réussir à coup sûr vos débuts sur le PCW. On découvre pas à pas le puissant taitement de texte LOCOSCRIPT, puis la programmation BASIC MALLARD et l'utilisation de CP/M. Indispensable pour bien profiter de son PCW.

Ref. : ML 164
Prix : **129** FF



Ref. : ML 165
Prix : **179** FF
Disponible en Juin

LE LIVRE DE L'AMSTRAD PCW

Vous possédez un PCW et vous voulez en tirer le maximum ? Alors ce livre a été écrit pour vous ! Grâce à lui vous utiliserez au mieux le LOCOSCRIPT et profiterez de toutes les possibilités offertes par le CP/M. Une formation intensive au BASIC MALLARD vous permettra d'écrire des routines d'édition, un générateur de masques de saisie, des routines de tri et une gestion de fichier.

les plus M.A. pour P.C.W.



NOUVEAU

dB Compiler : un compilateur pour dBASE II !

Le compilateur dB Compiler traduit votre programme écrit sous dBASE II en un jeu d'instructions proche du "langage machine" (donc très rapide) et pouvant être exécuté indépendamment de dBASE II. dB Compiler est le premier compilateur pour dBASE II, le leader des SGBD sur AMSTRAD PCW. dB Compiler permet de faire fonctionner vos applications dBASE II sans dBASE II. Celles-ci peuvent être diffusées librement sans aucune redevance à payer.

Les avantages de dB Compiler :
- Simplicité d'emploi
- Pas de redevance
- Protection du Code Source
- Indépendance de dBASE II.
- Accroissement de la vitesse d'exécution (jusqu'à 10 fois plus rapide !)
- Gestion entièrement automatique de la mémoire.
- Facilité de maintenance des programmes source.

Avec dB Compiler votre application fonctionnera plus rapidement et indépendamment de dBASE II. Vous pourrez ainsi la recopier et la faire fonctionner sans dBASE II.

Documentation en français.

DB Compiler
Ref AM312
Prix 790 F TTC



Les Best de Micro-Application



LE LIVRE DU CP/M AMSTRAD (Tome 12)

Ce livre vous permettra d'utiliser CP/M sur les CPC 464 664 et 6128 sans aucune difficulté. Vous y trouverez de nombreuses explications et les différents exemples vous assureront une maîtrise parfaite de ce très puissant système d'exploitation qu'est CP/M.
Ref ML 128
Prix **149** FF

Fonctionne sur PCW



BIEN DEBUTER AVEC LE PCW

Le premier livre pour l'AMSTRAD PCW ! Cet ouvrage vous permettra de réussir à coup sûr vos débuts sur le PCW. On découvre pas à pas le puissant taitement de texte LOCOSCRIPT, puis la programmation BASIC MALLARD et l'utilisation de CP/M indispensable pour bien profiter de son PCW.
Ref ML 164
Prix **129** FF



LE LIVRE DE L'AMSTRAD PCW

Vous possedez un PCW et vous voulez en tirer le maximum ? Alors ce livre a été écrit pour vous ! Grâce à lui vous utiliserez au mieux le LOCOSCRIPT et profiterez de toutes les possibilités offertes par le CP/M. Une formation intensive au BASIC MALLARD vous permettra d'écrire des routines d'édition, un générateur de masques de saisie, des routines de tri et une gestion de fichier.
Ref ML 165
Prix **179** FF



13 rue Sainte Cécile 75 009 PARIS
Tél (1) 47 70 32 44

BON DE COMMANDE

DESIGNATION	QUANTITE	PRIX
	TOTAL TTC	

CB date d'expiration: ____

☐ Mandat ☐ Chèque ☐ CCP

Libellez vos chèques à l'ordre de Micro-Application.

Nom, Prénom ____

Adresse ____

Ville ____ C.P ____

CB ☐☐☐☐ ☐☐☐ ☐☐☐☐ ☐☐☐

Date et signature

\+ 20 F de frais d'envoi
ou 40 F pour envoi recommandé
Port gratuit pour toute commande supérieure à 250 F

AM-DB-86

ACHETEZ LA DISQUETTE

Ce livre vous passionne, mais vous n'avez pas le temps de taper les programmes.

Commandez nous la disquette des programmes du livre :

LA BIBLE DU GRAPHISME SUR CPC

BON DE COMMANDE

NOM :

PRENOM :

ADRESSE :

CODE POSTAL : VILLE :

Je désire recevoir la disquette du livre :

LA BIBLE DU GRAPHISME SUR CPC

Je joins à ce coupon un [] chèque de : 120 FF [] CCP de : 120 FF

DATE : SIGNATURE :

MICRO APPLICATION

13 RUE SAINTE CECILE

75009 - PARIS

Achevé d'imprimer en janvier 1987
sur les presses de l'imprimerie Laballery
58500 Clamecy
Dépôt légal : janvier 1987
Numéro d'imprimeur : 701017

LA BIBLE DU GRAPHISME

Les graphismes sur ordinateur ! Ce thème est plus que d'actualité et tous les utilisateurs de micro-ordinateurs sont passionnés par cette application de l'informatique.

Mais jusqu'à maintenant, les possesseurs d'AMSTRAD CPC et PCW se sont sentis un peu désavantagés dans ce domaine.

Ce livre a pour objet de leur montrer comment profiter des capacités graphiques du CPC (et PCW), comment réaliser des effets d'animation spectaculaires ou du graphisme en 3 DIMENSIONS.

Avec ce livre, vous obtiendrez sur votre écran des images et des effets que vous croyiez impossible auparavant ou réservés à des ordinateurs plus puissants.

Extraits du contenu du livre :

- Pour apprendre les bases : un logiciel PAINT.
- Editeur de Caractères.
- CPC CHART : un Générateur de Graphiques.
- DESTROY : un Jeu d'Arcade.
- Traceur de Fonctions.
- Les graphismes Vectoriels.
- CPCs WORLD : un Logiciel d'Animation.
- Comment programmer les Graphismes en Langage Machine ?
- Listing Commenté de la ROM GRAPHIQUE.
- Les SPRITES (lutins) sur CPC et un Générateur de Sprites.

- Tout sur GSX pour le CPC 6128 et les PCW 8256/8512 avec GDOS commenté.

Tous les programmes de ce livre sont fournis en listing source et commentés.

Réf. : ML 181
Prix : 199 Francs
ISBN : 2-86899-090-8

STEIGERS
VERVEST

AMSTRAD
CPC

LA BIBLE DU GRAPHISME

MA

LA BIBLE DU GRAPHISME

AMSTRAD
CPC/PCW



EDITIONS MICRO APPLICATION

UN LIVRE DATA BECKER



Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/